UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de
Ingeniería y Sistemas de Telecomunicación

PROYECTO FIN DE GRADO

PLATAFORMA DE CONOCIMIENTO EVOLUTIVO PARA DETECCIÓN TEMPRANA DE TRASTORNOS DEL LENGUAJE

MIGUEL MENÉNDEZ ÁLVAREZ

Grado en Ingeniería Telemática

Julio 2014

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: Plataforma de conocimiento evolutivo para detección temprana de trastornos del lenguaje

AUTOR: Miguel Menéndez Álvarez

TITULACIÓN: Grado en Ingeniería Telemática

TUTORA: Mª Luisa Martín Ruiz

DEPARTAMENTO: Departamento de Ingeniería y Arquitecturas Telemáticas

VºBº

Miembros del Tribunal Calificador:

PRESIDENTE: Pilar Ochoa Pérez

TUTORA: Mª Luisa Martín Ruiz

SECRETARIO: Iván Pau de la Cruz

Fecha de lectura: 18 de julio de 2014

Calificación:

El Secretario,

Agradecer a Marisa, mi tutora en este proyecto, el esfuerzo y empeño puesto en toda esta plataforma…y por elegirme a mí.

Y a la familia que siempre está ahí…

Resumen

Este Proyecto Fin de Grado, es el primer paso para abordar la construcción de una plataforma de conocimiento evolutivo para dos sistemas que facilitan la detección precoz de trastornos del lenguaje en niños de 0 a 6 años. Concretamente, el objetivo principal de este proyecto es el diseño, desarrollo y puesta en explotación de un sistema de recogida de propuestas de mejora sobre la base de conocimiento de los sistemas de ayuda a la toma de decisiones Gades y Pegaso. Este sistema está formado fundamentalmente por una aplicación diseñada y construida mediante una arquitectura de componentes de software modular y reutilizable. La aplicación será usada por los usuarios de las plataformas Pegaso y Gades para realizar las propuestas de cambio sobre la base de conocimiento de dichos sistemas. El sistema es accesible vía web y almacena toda la información que maneja en una base de datos. Asimismo, expone un estudio de aplicaciones orientadas al trabajo colaborativo (CSCW) y a la toma de decisiones colaborativa, como paso previo al desarrollo de una funcionalidad futura del propio sistema.

Abstract

This Final Degree Project, is the first step to address the construction of a platform for two evolutionary knowledge systems that facilitate early detection of language disorders in children aged 0-6 years. Specifically, the main objective of this project is the design, development and start-up of a system that collect improvement proposals about the knowledge of decision support systems Gades and Pegaso. This system consists mainly of an application designed and built by a modular component architecture and reusable software. The application will be used by users of the Pegaso and Gades platforms for change proposals on the basis of knowledge of such systems. The system is accessible via web and stores all the information managed in a database. It also presents a study of collaborative work oriented applications (CSCW) and collaborative decision making, prior to the development of a future system functionality.

Índice de contenidos

Índice de figuras

Índice de Tablas

Lista de acrónimos

API	*Application Programming Interface* (Interfaz de Programación de Aplicaciones)
BC	Base de Conocimiento
BD	Base de Datos
CSCW	*Collaborative-supported Computer Work* (Trabajo colaborativo asistido por computadora)
CSS	*Cascading Style Sheets* (Hojas de Estilo en Cascada)
DAO	*Data Access Object* (Objeto de Acceso a Datos)
EJB	*Enterprise Java Beans* (Componente Java de Empresa)
HTML	*HyperText Markup Language* (Lenguaje de Marcas de Hipertexto)
HTTP	*HyperText Transfer Protocol (Protocolo de transferencia hipertextual)*
Java EE	*Java Enterprise Edition* (Java Edición Empresarial)
JDK	*Java Development Kit (*Kit de Desarrollo de Java)
OWL	*Web Ontology Language* (Lenguaje de Ontologías Web)
RDF	*Resource Description Framework* (Marco de Descripción de Recursos)
SQL	*Structured Query Language* (Lenguaje de Consulta Estructurado)
UML	*Unified Modeling Language* (Lenguaje Unificado de Modelado)
XML	*eXtensible Markup Language* (Lenguaje de Marcas Extensible)

Capítulo 1. Introducción

La detección de las posibles alteraciones en el desarrollo infantil es un aspecto fundamental de la atención temprana en la medida en que va a posibilitar la puesta en marcha de los distintos mecanismos de actuación de los que disponen las entidades implicadas. Cuanto antes se realice la detección, existirán mayores garantías de prevenir patologías añadidas, lograr mejoras funcionales y posibilitar un ajuste más adaptativo entre el niño y su entorno.

El presente Proyecto Fin de Grado supone una mejora, desde el punto de vista funcional, de las plataformas web de e-Salud Pegaso y Gades. Dichas plataformas han sido caracterizadas, modeladas y descritas por la tutora del presente proyecto, Mª Luisa Martín Ruiz, como parte de su tesis doctoral.

Tomando como base las primeras versiones desarrolladas en el marco de la citada tesis, ambos sistemas fueron mejorados en el Proyecto Fin de Grado titulado "*Implementación de una arquitectura de componentes para un sistema de apoyo a la toma de decisiones en Atención Primaria*" [URI13].

Cada una de estas plataformas constituye un servicio telemático cuya finalidad es tratar de facilitar, a los profesionales que trabajan con poblaciones infantiles entre los 0 y 6 años, la detección precoz de trastornos del lenguaje en niños. Cada plataforma está orientada a un ámbito concreto, siendo ambas constitutivas de **sistemas de apoyo a la toma de decisiones**.

Mientras que Pegaso se orienta hacia el personal médico (pediatras, neuropediatras, Atención Temprana, etc.), Gades está enfocada hacia los colegios y los educadores infantiles. Ambas tienen en común que se apoyan en una base de conocimiento desarrollada previamente.

Esta base de conocimiento debe contener la información necesaria para, mediante un mecanismo sencillo, permitir realizar evaluaciones del lenguaje en un tiempo reducido.

La base de conocimiento está formada por una serie de tuplas como las que recoge la Tabla 1.

Tabla 1: Ejemplo extraído de la base de conocimiento

Hito	Descripción	Decisión Sistema
33 meses - Aviso	Nombra 4 imágenes	Adelantar visita (en un mes)
33 meses - Aviso	Indica su edad con los dedos	Adelantar visita (en tres meses)
33 meses - Aviso	Conoce 2 acciones	Adelantar visita (en tres meses)
33 meses- Aviso	Cuenta un bloque	Adelantar visita (en tres meses)
33 meses - Aviso	Se entiende la mitad de lo que habla	Adelantar visita (en tres meses)

La columna Descripción recoge la pregunta que el sistema presenta al usuario (pediatra, educador, logopeda) con objeto de evaluar el estado de adquisición del lenguaje en el niño ("Nombra 4 imágenes"). Esta descripción está relacionada con una edad a la que el niño debe haber adquirido esas destrezas (33 meses) y el tipo de hito que constituye la no superación del mismo ("Aviso" o "Alarma", siendo esta última más grave). Por último, en caso de no obtenerse una respuesta adecuada se indicará al usuario la decisión que recoge la columna Decisión Sistema.

Este proyecto surge de la necesidad de **evolucionar el conocimiento** recogido en las plataformas Pegaso y Gades. Esta evolución supone mejorar la base de conocimiento de cada una de ellas, de forma que las evaluaciones realizadas sean más correctas tanto en las cuestiones que se planteen como en las recomendaciones del sistema. Por otro lado, esta evolución debe realizarse fruto de la colaboración y el consenso entre expertos y generándose de forma automática, es decir, sin la necesidad de intervención por parte de un ingeniero.

Esta evolución estaría dividida en dos fases: realización de propuestas y decisión consensuada de cambios.

Una primera fase en la que tanto los usuarios de dichas plataformas (pediatras, logopedas, educadores, etc.) como el propio sistema colaboran realizando o proponiendo diversos cambios en la base de conocimiento. Los usuarios están involucrados en el uso del sistema y pueden observar con mayor precisión los posibles problemas, siendo éstos la principal fuente de propuestas. Otra opción contemplada consiste en que el propio sistema podría realizar propuestas de cambio en base a los datos recogidos y a las estadísticas obtenidas, ya que podría observar problemas mucho más generales que un solo usuario, por ejemplo, una pregunta que un 100% de los usuarios resuelve igual, bien sea una respuesta positiva o negativa, es muy probable que sea una pregunta parcialmente errónea. El sistema puede detectarlo, pero para un usuario concreto puede no ser representativo. Lo mismo ocurre si un alto porcentaje

de usuarios es incapaz de responder a una pregunta (respuesta NS/NC). La finalidad de las preguntas es detectar comportamientos anómalos y la existencia de una pregunta de la que siempre se obtiene la misma respuesta resulta inútil para el sistema.

Una segunda fase en la que un grupo de expertos utilizarían esas propuestas recogidas en la primera fase para decidir los diferentes cambios que debe sufrir la base de conocimiento. Para ello deberán llegar a un acuerdo consensuado, ya que la sensibilidad del tema tratado requiere un consenso más allá de una votación.

Cada una de estas fases supone uno de los objetivos de este proyecto. Por un lado, la necesidad de obtener propuestas por parte de los usuarios deriva en la creación de una plataforma web que las recoja. Y, por otro lado, el consenso entre expertos supone la realización de un estudio y análisis sobre el trabajo y la toma de decisiones de manera colaborativa.

Por último, el sistema debe ser capaz de modificar la base de conocimiento una vez aprobados los cambios por los expertos. La base de conocimiento está representada por una ontología, que no es más que un conjunto de reglas que permiten describir y representar el conocimiento de un determinado contexto o dominio. En la práctica, lo único que hacen estas reglas es definir conceptos básicos y las posibles relaciones entre ellos. A su vez, para representar esta ontología en el contexto web utilizamos OWL (Ontology Web Language). La utilización de este lenguaje supone la generación de un fichero OWL con la ontología, el cual será utilizado por Pegaso y Gades a la hora de realizar evaluaciones del lenguaje. Su creación automática supone el último paso para obtener una plataforma sin la participación de personal técnico.

1.1. Objetivos

Los objetivos del presente proyecto vienen dados por la necesidad de evolución de las plataformas Pegaso y Gades antes mencionada, siendo específicamente:

- Realizar un estudio y análisis de las plataformas existentes en cuanto a **CSCW (Computer-supported Cooperative Work)** y **toma de decisiones colaborativa** aplicados en entornos de e-Salud.
- Realizar un estudio y análisis de la **creación dinámica/automática de una ontología válida** que permita la automatización total de la evolución de la base de conocimiento.
- Crear una base de datos que represente la base de conocimiento a partir de la ontología existente.

- Desarrollar una plataforma web usable que permita, a los usuarios de Pegaso y Gades, la **realización de propuestas** de cambio que debe recoger para la base de conocimiento.

1.2. Estructura

La presente memoria describe el trabajo realizado en el curso del Proyecto y está organizada en varios capítulos cuyos contenidos se detallan a continuación:

- **Capítulo 1: Introducción.** Explica la motivación y los objetivos de este Proyecto Fin de Grado, así como la estructura del mismo.
- En el **capítulo 2, Antecedentes y marco tecnológico**, se expone una visión general del estado actual del campo en el que se encuadra este trabajo, así como una descripción detallada de los antecedentes y del estado actual de los sistemas de partida.
- En el **capítulo 3, Descripción de la solución propuesta,** se describen las soluciones propuestas a los diferentes problemas planteados en el proyecto. Se comienza con una descripción general de las propuestas, para después dar paso a los detalles de cada uno de los aspectos de las soluciones, desgranando el proceso de desarrollo seguido y justificando las decisiones de diseño tomadas.
- En el **capítulo 4, Resultados,** se detallan todas las pruebas a las que han sido sometidas las soluciones propuestas en el capítulo anterior para comprobar su correcto funcionamiento. Asimismo, se incluyen los resultados obtenidos en dichas pruebas de manera sintetizada.
- En el **capítulo 5, Conclusiones,** se expone una visión general del trabajo realizado y expuesto en los capítulos anteriores, y se plantean las conclusiones derivadas del mismo.
- En el **capítulo 6, Trabajo futuro,** se proponen una serie de líneas de trabajo futuras que, bien por escasez de tiempo o por estar fuera del alcance de este proyecto, no han sido abordadas.
- En el **capítulo 7, Referencias,** se encuentran las fuentes bibliográficas empleadas durante el desarrollo del presente trabajo.

Capítulo 2. Antecedentes y marco tecnológico

Los antecedentes inmediatos se encuentran en los proyectos "*Implementación de una arquitectura de componentes para un sistema de apoyo a la toma de decisiones en Atención Primaria*" [URI13] y "*Modelo de conocimiento para detección precoz de trastornos del lenguaje en niños de 0 a 6 años*" [MAR11] realizados por Javier Martín Uría y Mª Luisa Martín Ruiz (tutora de este proyecto) respectivamente, y que suponen el punto de partida y la motivación del presente proyecto.

El trabajo de investigación desarrollado por Mª Luisa Martín Ruíz recoge la conceptualización de los sistemas Pegaso y Gades. Para ello se presenta el modelo de conocimiento en el cual estarán basadas ambas plataformas. Asimismo, inicia la implementación de la plataforma web, elemento retomado por Javier Martín Uría y culminado con la finalización de una plataforma completamente operativa.

El sistema Pegaso tiene como objetivo permitir a pediatras de atención primaria la realización de evaluaciones del lenguaje a niños de entre 0 y 6 años. Estas evaluaciones suponen un apoyo en la toma de decisiones para sus usuarios, beneficiándose del conocimiento del sistema y de las decisiones propuestas. Del mismo modo, ofrece un mecanismo de gestión de usuarios mediante el cual poder manejar las diferentes altas y bajas de médicos y especialistas. Por último, constituye una base de datos relacional que permite mantener todos los datos de las evaluaciones, pacientes y médicos, pudiendo consultar éstos siguiendo diferentes tipos de filtros (médico, paciente, etc.).

Por su parte, el sistema Gades permite a los profesionales que trabajan en escuelas infantiles la identificación de trastornos del lenguaje en sus alumnos. Al igual que Pegaso, permite realizar evaluaciones del lenguaje a niños de entre 0 y 6 años, pero con la diferencia de que las decisiones del sistema no serán definitivas. Estos resultados serán tratados por el colegio pudiendo tomar diferentes medidas como realizar un seguimiento al niño o recomendar ciertas acciones a los padres (derivar al neuropediatra, al equipo de atención temprana).

El conjunto de ambos proyectos supone el punto de partida, ya que marca las pautas a seguir en aspectos como el diseño de la plataforma web o la base de datos como detallaremos más adelante.

Por otro lado, uno de los objetivos del presente proyecto es el estudio y análisis de las plataformas actuales de CSCW (*Computer-Supported Cooperative Work*) y toma de decisiones involucradas en entornos de e-Salud. Este estudio explica en qué consiste cada una de estas tecnologías, describiendo posteriormente algunos ejemplos de plataformas que las emplean y que están relacionadas con este proyecto.

2.1. Trabajo cooperativo asistido por computadora

En primer lugar, CSCW son las siglas en inglés de "trabajo cooperativo asistido por computadora". El CSCW fue utilizado por primera vez en 1984 por Irene Greif, del Massachussetts Institute of Technology, y Paul Cashman, de la Digital Equipment Corporation. En ese año tuvo lugar en Estados Unidos el primer taller de CSCW, cuyo objetivo fue reunir a personas de diferentes disciplinas para analizar las características de comunicación y coordinación en grupos de trabajo.

Fue en 1984 cuando, debido sobre todo a un enfoque del trabajo en grupos, ocurrió la llamada "automatización de oficinas", la cual cristalizó en un esfuerzo para integrar las aplicaciones de un solo usuario para ser compatibles con grupos. Pero en la automatización de oficinas no se tenía el conocimiento necesario para entender cómo trabaja la gente en grupos y organizaciones y cómo la tecnología afecta a su trabajo, lo que trajo consigo un interés hacia el estudio del CSCW por parte de economistas, psicólogos sociales, antropólogos teóricos y organizacionales, educadores y muchos otros que podían arrojar luz sobre las actividades grupales.

Las conexiones entre diferentes comunidades y paradigmas hace necesario llevar a cabo investigaciones que faciliten la comprensión de la complejidad y los matices de CSCW. Es necesario evaluar el impacto de CSCW en las organizaciones; puesto que se han hecho muchos esfuerzos para su diseño, pero muy pocos para la evaluación de estas tecnologías.

Es importante medir la eficiencia de un sistema cooperativo puesto que implica un alto coste en tiempo y dinero. También porque existen diversidad de necesidades por las que esta tecnología puede ser empleada en las empresas, y que podrían conducir a la investigación de formas diferentes e innovadoras de usarla.

El estudio realizado muestra la existencia de múltiples soluciones desarrolladas para simplificar y mejorar el rendimiento de las tareas que requieren un trabajo colaborativo en entornos relacionados con la salud. Estas herramientas tienen objetivos concretos y diferentes en función de los problemas que deben resolver

2.1.1. Plataforma CSCW: Teleworks

Teleworks es una aplicación para la ayuda al diagnóstico médico y a la teleconsulta. El sistema ofrece una plataforma donde los médicos pueden intercambiar información (desde lugares diferentes), orientado a la realización de llamadas de voz y videollamadas. *Teleworks* permite compartir documentos y editarlos en tiempo real, de manera que podrían compartir pruebas gráficas, como una radiografía, y realizar diferentes indicaciones y comentarios a la misma, colaborando así para una interpretación más correcta.

En la Figura 1 podemos observar un ejemplo en el que se observa una imagen médica compartida en la cual se ha señalado una zona y se añade un comentario textual.

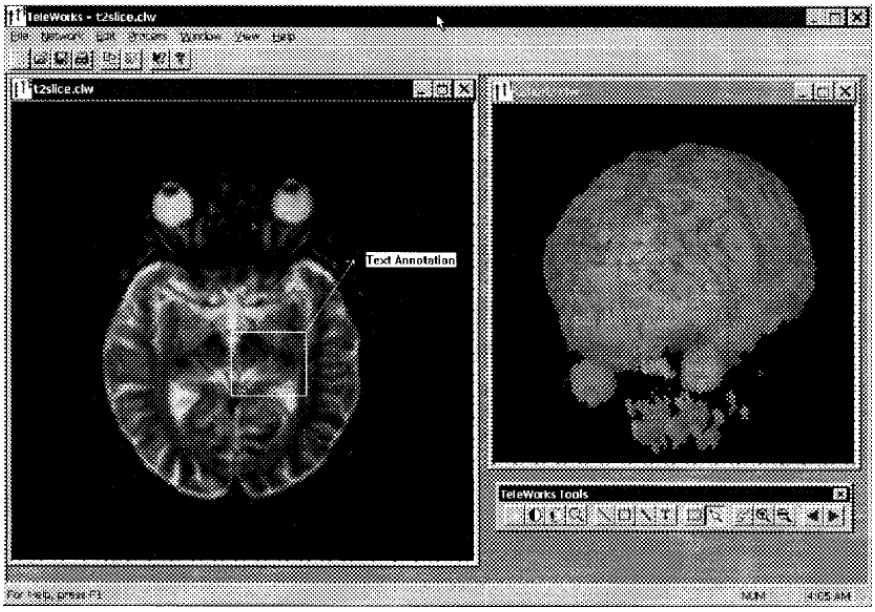


Figura 1: Pantalla principal de Teleworks

Este servicio incluye garantías de seguridad como autenticación, asegurando que un usuario es quién dice ser; control de acceso, impidiendo que entidades no autorizadas accedan a los recursos; confidencialidad, protegiendo los datos de la revelación a un usuario no autorizado; integridad, garantizando que los datos recibidos coinciden con los enviados (detecta añadidos, sustracción o cambios); y no repudio, evitando que emisor o receptor puedan negar la transmisión de un mensaje. Todo ello debido a la sensibilidad de los datos y a la obligada protección de los mismos.

2.2. Toma de decisiones colaborativa

En segundo lugar, la toma de decisiones colaborativa (también conocido como *collaborative decisión-making*) forma parte de las herramientas CSCW, ya que implica la colaboración de un grupo de personas utilizando diferentes herramientas tecnológicas a la hora de realizar un trabajo, en este caso, tomar una decisión.

En la toma de decisiones colaborativa varios individuos deben tomar una decisión entre una serie de alternativas existentes. Las decisiones tomadas por los grupos suelen ser diferentes de las realizadas por los individuos, ya que participan un mayor número de personas y, por lo tanto, de ideas, conocimientos, creencias, etc. que influyen en el proceso de toma de decisiones.

Hay un gran debate en cuanto a si esta diferencia se traduce en decisiones que son mejores o peores. De acuerdo con la idea de sinergia, las decisiones tomadas colectivamente tienden a ser más eficaces que las decisiones tomadas por un solo individuo. Sin embargo, también puede darse que las decisiones tomadas por un grupo sean erróneas.

Para tratar de evitar estos problemas se han propuesto diversos sistemas que los resuelven de diferentes formas.

El primero es el consenso en la toma de decisiones, este método trata de evitar los "ganadores" y "perdedores". El consenso requiere que una mayoría apruebe un determinado curso de acción, pero que la minoría esté de acuerdo en continuar el curso de acción. En otras palabras, si la minoría se opone a la línea de acción, el consenso requiere que el curso de acción se modifique para eliminar características objetables.

En segundo lugar, encontramos los métodos basados en votaciones que permiten a cada miembro otorgar una puntuación a una o varias de las opciones disponibles de donde se elige la opción con el promedio más alto. La mayoría requiere el apoyo de más del 50% de los miembros del grupo. Por lo tanto, la exigencia es menor que con la unanimidad y un grupo de "perdedores" está implícito a esta regla. Otra opción es la pluralidad, donde el bloque más grande de un grupo decide, incluso si no llega a la mayoría.

Todos ellos no suponen una solución infalible, sino que pretenden minimizar los aspectos negativos de las decisiones tomadas en grupo y potenciar los positivos. Pese a

existir más métodos con este fin, en este estudio solo se recogen las aproximaciones más significativas.

De este modo surgen herramientas orientadas a automatizar estas tareas y a resolver los problemas que puedan generar. Estas herramientas reciben el nombre de *Collaborative Decision Support Systems* (CDSS). Para este grupo encontramos dos proyectos muy similares a los requerimientos de nuestro sistema.

2.2.1. Hermes

El proyecto descrito en el artículo titulado *"Computer supported argumentation and collaborative decision making: the Hermes system"* [**KAR99**], se encuadra dentro del ámbito de la toma de decisiones en el área de la salud. La función principal del sistema Hermes es facilitar la discusión sobre un tema y organizar el curso de la misma, de manera que sea fácil de seguir para los participantes y a su vez lo suficientemente compleja para representar todos los posibles escenarios que se puedan dar.

El sistema está basado en la generación de diagramas que ofrecen un seguimiento de las discusiones. Cada discusión se organiza en base a unas etiquetas de "asunto" (*issue)*, dentro de las cuales existen:

- Alternativas (*alternative*),
- Restricciones (*constraint*),
- Posiciones a favor y en contra (*position in favor/against*).

Esto permite organizar la discusión y mantener una vista del estado de la misma. La Figura 2 muestra estas etiquetas con sus respectivos iconos.

Figura 2: Notación utilizada por Hermes

De una manera más detallada y con la ayuda de la Figura 3 podemos observar que cada entrada en el foro de discusión se corresponde con un elemento de la argumentación.

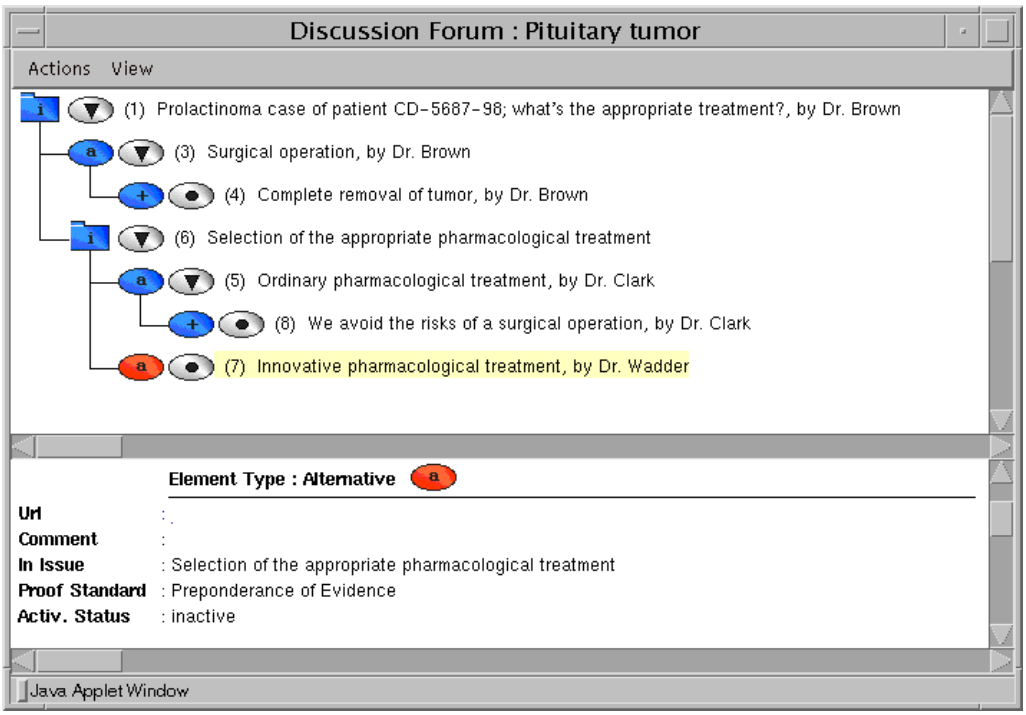


Figura 3: Ventana ejemplo del sistema Hermes [KAR99]

Cada elemento está acompañado por dos iconos:

- Uno que indica el tipo de elemento y
- Otro que sirve para plegar/desplegar (*Folded/Unfolded item* o *Nothing to fold/unfold* de la **¡Error! No se encuentra el origen de la referencia.**).

Por otra parte, los números que aparecen entre paréntesis (junto a los iconos) corresponden a un identificador único del elemento en la base de datos (IDs que se asignan de forma automática). Por último, cada entrada en el foro puede contener el nombre de usuario que lo envió y la fecha.

El panel inferior de la ventana proporciona toda la información necesaria acerca de una entrada seleccionada en la gráfica de discusión (por ejemplo, los usuarios pueden seleccionar una entrada haciendo clic sobre la misma).

Los asuntos (issues) se corresponden con las decisiones que se deben tomar, o las metas que se deben alcanzar, son la raíz de las entradas del foro y son creados por los usuarios. Los diferentes asuntos consisten en un conjunto de alternativas que corresponden a las opciones posibles (por ejemplo, alternativa-3: "operación quirúrgica", y alternativa-5: "El tratamiento farmacológico", ambas pertenecen al asunto-1, y se han propuesto por el Dr. Brown y el Dr. Clark, respectivamente).

Si se quiere agrupar alternativas dentro de un asunto, Hermes permite crear un nuevo asunto dentro de uno de los temas y colocar ahí las alternativas (ver Figura 4). Por ejemplo, supongamos que hay dos tratamientos farmacológicos alternativos, uno ordinario y otro innovador.

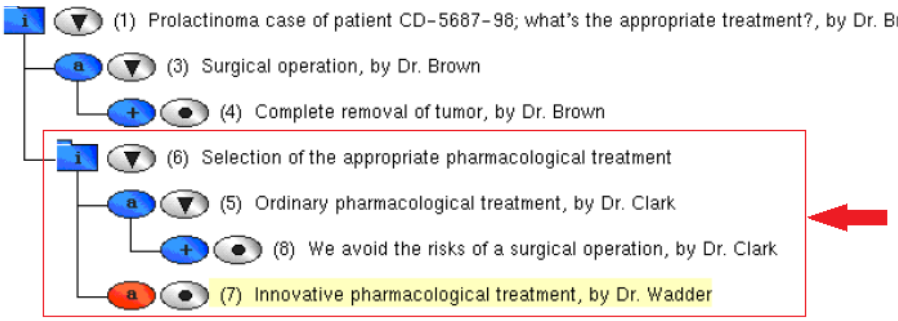


Figura 4: Subasunto agrupando alternativas en Hermes

Las posiciones (a favor o en contra) se utilizan para apoyar la selección de un curso de acción específico (alternativa), o para expresar una objeción a dicha alternativa de forma que se recoja la falta de interés de los usuarios en la misma. Por ejemplo, en la Figura 5, la posición-4: "La eliminación completa del tumor" sirve para apoyar la alternativa-3, mientras que la posición-9: "Este tratamiento va a durar mucho tiempo" sirve para expresar la objeción del Dr. Brown a la alternativa-5. Las posiciones también pueden referirse a alguna otra posición con el fin de proporcionar información adicional al respecto.

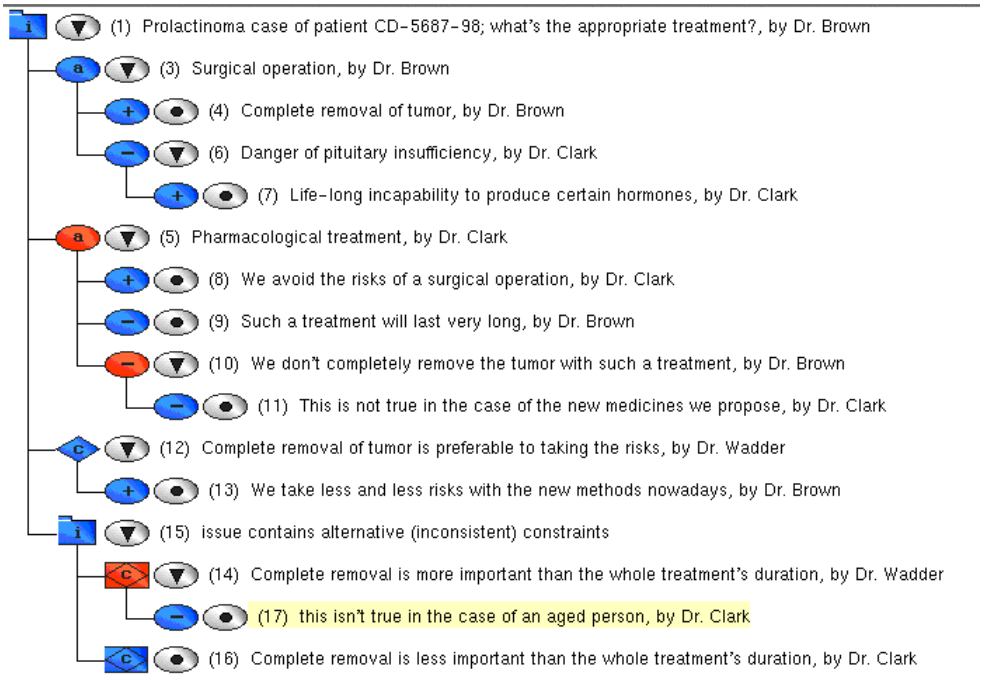


Figura 5: Discusión mediante el sistema Hermes

Una posición (a favor o en contra) **se refiere a** otra posición o alternativa, mientras que una alternativa **pertenece** siempre a un solo asunto. En los entornos de toma de decisiones, a menudo se tienen que definir prioridades entre las acciones y sopesar diferentes criterios de actuación.

En Hermes, las restricciones proporcionan una forma cualitativa de sopesar razones a favor y en contra de la selección de un determinado curso de acción. Una restricción es una combinación de los valores de dos posiciones y su relación de preferencia, donde esta relación puede ser "más/menos importante que" o "de igual importancia que" (ver Figura 6). Las restricciones se refieren a un asunto (o un subasunto interno) y pueden dar diferentes niveles de importancia a las alternativas. Al igual que los otros elementos de la argumentación, son objeto de discusión; por lo tanto, pueden estar "vinculados" con otras posiciones de apoyo o de desafío. En la Figura 5**¡Error! No se encuentra el origen de la referencia.**, la restricción-12: "La liminación completa del tumor es preferible a tomar riesgos" expresa la relación de preferencia " la posición-4 es más importante que la posición 8", se refiere al asunto-1 y se apoya en la posición 13.

Otras funcionalidades del sistema:

Además de las herramientas básicas comentadas para organizar las discusiones, el sistema Hermes dispone de utilidades que suponen una ayuda automática para la correcta realización de dichas discusiones.

Los estándares de prueba (Proofs standards)

Permiten definir si una alternativa, posición o restricción tiene el estado de *activa* o *inactiva*. Estos estados se actualizan mediante el estudio que realiza sobre el desarrollo de la argumentación y el sometimiento a diferentes estándares de prueba.

El sistema contempla varios estándares al tener en cuenta que no todos los elementos, ni siquiera de una misma argumentación, necesitan el mismo tipo de evidencia. Por ello, utilizan los siguientes tipos de evidencia:

- *Chispa* de evidencia (*Scintilla of Evidence*), donde una posición estará activa si una posición activa argumenta a su favor.
- *Más allá de la duda (Beyond Reasonable Doubt)*, en la cual es suficiente que no existan posiciones activas que argumenten en contra.

- Preponderancia de la evidencia (*Preponderance of Evidence*), donde el peso de las posiciones activas a favor debe superar el de las posiciones contrarias.

Por otro lado, Hermes permite detectar conflictos e inconsistencias los cuales se indican mediante la etiqueta "consistencia" que utilizan las restricciones.

Cada vez que una restricción se inserta en el gráfico de discusión (ver Figura 6), el sistema comprueba si existen ambas posiciones constitutivas de la nueva restricción en otra restricción. En caso afirmativo, la nueva restricción se considera ya sea redundante, si también tiene la misma relación de preferencia, o en conflicto.

Una restricción redundante se ignora, mientras que una conflictiva figura, junto a la restricción introducida previamente en un tema creado automáticamente por el sistema, para reunir las restricciones conflictivas y estimular aún más la argumentación sobre ellas hasta que sólo una se vuelva activa.

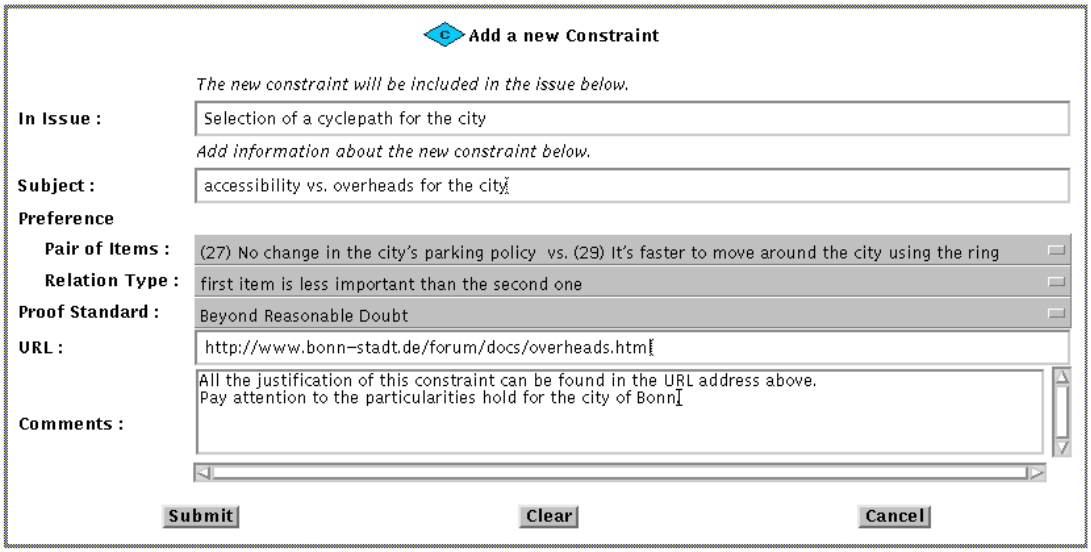


Figura 6: Ejemplo de añadir una nueva restricción en Hermes

Cabe señalar aquí que ambas posiciones constitutivas de una nueva restricción ya han sido insertadas en el gráfico de la discusión; por lo tanto, siempre que un usuario está a punto de insertar una nueva restricción, el sistema proporciona una lista de todas las combinaciones posibles para que el usuario seleccione una, como se observa en el apartado "*Pair of items*" de la Figura 6.

El sistema de pesos

Es otro de los mecanismos facilitados por Hermes. Parte de la idea de que el peso de una posición se incrementa cada vez que esa posición es más importante que otra

(y disminuye cuando es menos importante), con el objetivo de extraer un orden total de alternativas. Dado que sólo puede dar una información parcial, la elección de los pesos máximos y mínimos iniciales puede afectar a la recomendación del sistema.

Este esquema de ponderación no es la única solución; el artículo [KAR99] plantea la posibilidad de ejecución de esquemas alternativos, basados en diferentes algoritmos (el artículo no da más información de estos algoritmos, puesto que en ese momento se estaban implementando).

Búsquedas de información

Además, la plataforma permite realizar búsquedas de información sobre los registros existentes, lo que facilita el acceso a la información en casos futuros.

2.2.2. Collaborative Decision Making framework

El otro proyecto comentado anteriormente esta descrito en el artículo *"Collaborative Decision Making framework for Multi-Agent System"* **[IND08]**, que detalla una plataforma de toma de decisiones multiusuario. Esta plataforma se basa en una estructura modular en la que se reparten las diferentes tareas y funciones. Para nuestro estudio el módulo más importante es el relacionado con la toma de decisiones (Decision Making Module).

También son importantes los mecanismos empleados para la toma de decisiones grupal y cómo permite llevarla a cabo basándose en el cálculo de "valores de confianza" (*Trust Computation,* descrito a continuación).

Módulos del sistema

Esta plataforma está organizada en módulos, donde cada uno de ellos se encarga de diferentes partes del proceso. La siguiente figura muestra los módulos que conforman el sistema y las relaciones entre ellos:

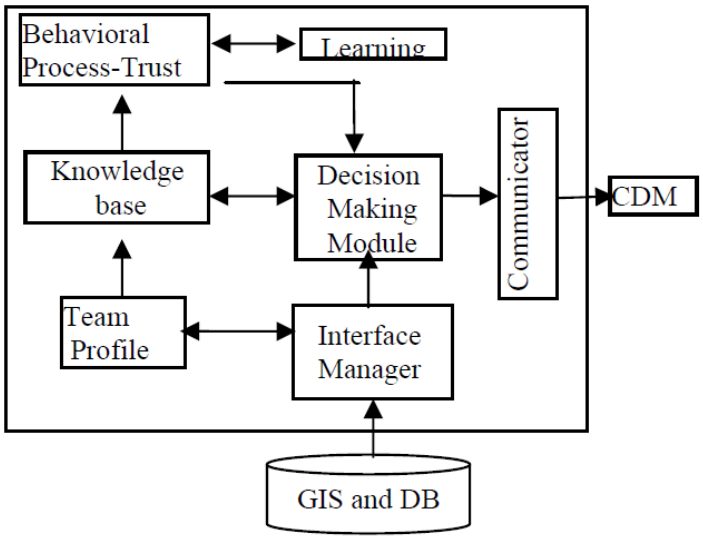


Figura 7: Sistema de toma de decisiones colaborativo

El primero de ellos es el **gestor de interfaz** (Interface Manager): se trata de un módulo que proporciona una interfaz gráfica para la base de datos de la aplicación. Este módulo extrae la información y los datos necesarios para realizar la tarea.

Después nos encontramos el módulo de **toma de decisiones** (Decision Making Module). Desde el gestor de interfaz se recogen los siguientes datos:

- Un conjunto de atributos, representando parámetros orientados a tareas (temas);
- Un conjunto de valores posibles para cada atributo (decisiones posibles para cada tema);
- Un conjunto de restricciones para los valores que pueden tomar, en función de los datos de entrada y de la tarea, puede ser nulo o tener algún valor.

El módulo de toma de decisiones utiliza los valores posibles y las restricciones para facilitar una solución automática del problema. Estas decisiones se pondrán a disposición de todos los agentes en el equipo.

El módulo de **perfil del equipo** (Team Profile): contiene información sobre cada agente, como nombre, dominio, creencias, y la reputación general de un agente (determinada por la organización y la decisión de la tarea por parte del agente). Los detalles del equipo están disponibles para cada agente a través de este perfil.

El módulo con la **base de conocimiento** (Knowledge base): la experiencia del agente se almacena como conocimiento. Se trata de un conjunto de hechos y reglas que

registra las últimas interacciones de un agente con otro (dentro el grupo). Estos datos cambian de forma dinámica con cada interacción del agente para reflejar el comportamiento del mismo hacia otros agentes.

El módulo de **aprendizaje** (Learning) en el que cada agente utiliza un aprendizaje gradual de la actividad. Durante la colaboración, la nueva información aprendida se representa en este módulo. El módulo de aprendizaje actualiza la información de la base de conocimiento.

El módulo de **comunicación (**Communicator) tiene una gran importancia. Este módulo de programa proporciona servicios de comunicación que permiten a un agente el intercambio de información con otros agentes. También registra todas las interacciones entre los agentes que utilizan el sistema de mensajes.

El módulo de **proceso de la conducta** (Behavioural Process-Trust) utiliza parámetros sociales de confianza. La experiencia directa y/o la interacción social de un agente entre el equipo juegan un papel importante en la determinación de la confianza en otros agentes. Proponen un modelo de confianza que contiene la experiencia directa de un agente con otros agentes, así como las recomendaciones de otros agentes. Este modelo de confianza ayuda a los agentes a colaborar en la tarea de llegar a un consenso y en la selección de una decisión final.

Por último, está el módulo de **toma de decisiones colaborativa (**CDM). El consenso en la toma de decisiones significa que todos los miembros están de acuerdo en que la decisión es aceptable. Este sistema propone un método de consenso basado en la *teoría del juicio social.* Este método sugiere que las diferencias entre los juicios individuales se producen debido a la diferencia en la importancia que conceden a la información y cómo se relacionan con la información para su juicio. Esta teoría intenta observar la estructura subyacente de una decisión individual y proporcionar esta información como retroalimentación a los participantes, utilizando el módulo comunicador.

Proceso de toma de decisión grupal

En este caso, el proceso de toma de decisión colaborativa es un proceso de varias etapas. Los expertos/agentes utilizan los modelos de decisión disponibles y experiencia en el tema, toman la decisión de la tarea y la aportan al sistema utilizando el módulo de toma de decisiones y el gestor de interfaz. Las decisiones individuales proporcionadas por los agentes deben ser evaluadas y la decisión final será

seleccionada en el proceso de decisión de grupo. Se permite a cada agente discutir con otros participantes y aclarar información sobre cada decisión utilizando el módulo de comunicación. Este proceso de decisión en grupo se describe en la Figura 8.

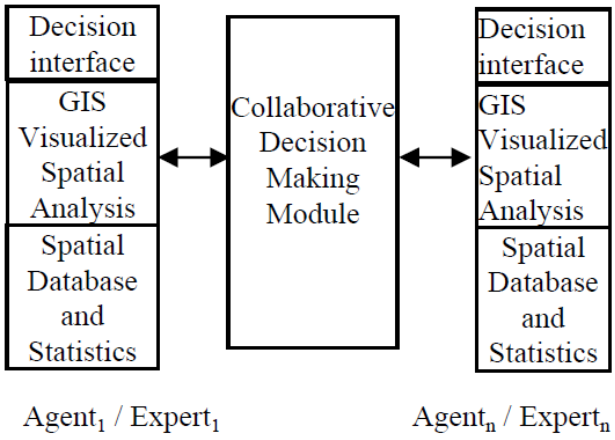


Figura 8: Arquitectura de agentes de colaboración para la toma de decisiones

En este proceso grupal de toma de decisiones, los agentes realizan en paralelo una serie de tareas. La primera de ellas es que reciben la información sobre la tarea a resolver y el equipo encargado. A continuación, procesan la información y aplican el modelo de decisión multicriterio, produciendo como salida un conjunto de decisiones hacia el módulo de toma de decisiones colaborativa (Collaborative Decision Making Module).

Una vez que el módulo ha recogido todas las decisiones de los agentes comienza la colaboración: se intercambia información; cada agente discute sobre cada decisión, criterio o conflicto; se actualizan los parámetros de comportamiento mediante la llamada *Trust Computation* ("computación o procesamiento de la confianza o nivel de verdad"), que obtiene información sobre la formación y evolución de este nivel de "verdad". Realizadas estas acciones, se agregan los valores de verdad y se elige como solución final la decisión con un valor de verdad más alto.

La confianza tiene muchas definiciones que son aplicables a las diferentes áreas de los sistemas de computación y sistemas distribuidos. Estas definiciones generalmente tratan de transmitir que esa confianza tiene algún valor cuantitativo asociado, es decir, una cantidad de confianza de un agente en otro basándose en la experiencia o la interacción social directa. En esto se basa el *Trust Computation* que realiza la plataforma, y que se describe a continuación.

Trust Computation

Como se comentaba anteriormente, esta fase está formada por dos etapas que se apoyan en mecanismos de menor envergadura, los cuales serán explicados a continuación.

- **Formación de la confianza**: inicialmente, puede ser calculada mediante la confianza directa o indirecta. La confianza directa se puede formalizar a través de interacciones sociales como la familiaridad y la creencia de similitud como se indica más adelante. Mientras que la confianza indirecta puede ser determinada por las recomendaciones dadas por los agentes acerca de otros agentes y la reputación general que figuran en el perfil del equipo de organización.
- **Evolución de la confianza**: durante la colaboración entre agentes, éstos actualizan sus valores de confianza dinámicos en base a las interacciones y la discusión respecto a las decisiones.

Conociendo las etapas del procesamiento de la confianza del sistema, procedemos a describir los diferentes parámetros que emplean y en qué se basan:

1. **Confianza Directa** (DT): contiene el nivel de confianza directa de un agente concreto en otros agentes. La confianza directa se puede formalizar a través de las interacciones sociales, como la experiencia previa directa en el trabajo en grupo, el correo electrónico, las consultas sobre cuestiones generales, las consultas sobre temas específicos del dominio tratado, etc. Esto se puede formalizar a través de la familiaridad y la similitud de la siguiente manera.
    - Familiaridad: refleja la observación del agente a través de las interacciones sociales y las consultas. La familiaridad se formalizará a través de la observación del número total de interacciones y del número total de interacciones exitosas y satisfactorias en un plazo determinado. El marco de tiempo debe ser lo suficientemente grande como para permitir llevar acabo las interacciones.
    - Creencias Similares: refleja la similitud de conocimiento del dominio por parte de dos agentes. Este valor se refiere a la similitud de las creencias de dos agentes en un dominio concreto, si dos agentes tienen creencias similares, el recuento similitud es superior al umbral T, entonces el valor de similitud es 1 o, en caso contrario, el valor es 0. Se puede calcular esta similitud para todos los dominios que comparten dos agentes. La similitud final entre agentes es la suma del valor para cada dominio.

De ahí que la Confianza Directa pueda calcularse como:

Confianza Directa = Familiaridad + Similitud de creencias

2. **Recomendaciones (R):** cada agente calcula la recomendación sobre otro agente, sumando la confianza directa (DT) y la reputación general (GR) del perfil del equipo de acuerdo con la siguiente ecuación:

   *Recomendación = peso1*Confianza Directa + peso2*Reputación general*

   Donde peso1 y peso2 son la preferencia subjetiva del agente en cuanto a qué valor le aporta más relevancia. Cada agente durante la colaboración deberá intercambiar su recomendación con los otros agentes del grupo.

3. **Confianza en Experto del Dominio (DET):** evalúa la calidad de los conocimientos del dominio de un agente. El DET se refiere a la estimación de un agente acerca de la calidad de las creencias de otro agente. El valor debe estar entre 0 y 1. Cada agente, durante la fase de colaboración, deberá intercambiar su recomendación sobre otros agentes del grupo.

Durante la fase de colaboración, los valores de confianza directos y de recomendación se actualizan dinámicamente. Una vez que todos tienen los valores de cada agente evalúan las decisiones dadas por los agentes basándose en los valores de verdad obtenidos.

La confianza final en todas las decisiones dictadas por todos los agentes del grupo se calcula agregando todos los valores (DT, R y DET) de cada una para seleccionar la decisión final como la decisión con mayor valor de verdad.

Estos sistemas suponen el punto de partida a la hora de resolver la segunda fase de la evolución comentada en la introducción: el consenso entre expertos. Tras la recogida de propuestas de cambio en la plataforma Galatea creada en el presente proyecto, es necesario desarrollar una nueva funcionalidad en Galatea que permita la evaluación de las propuestas generadas mediante un sistema de toma de decisiones colaborativa basado en las ideas expuestas que no es alcance de este proyecto.

2.3. Sistemas Gades y Pegaso

Gades y Pegaso son los antecedentes inmediatos del sistema Galatea desarrollado en el presente proyecto, además de ser la justificación principal del mismo.

La existencia de estos proyectos previos y sus características tiene una gran relevancia para el sistema, siendo el motivo de la elección de plataforma de desarrollo o de la propia arquitectura software del sistema.

Por ello, a continuación se detalla:

- Plataforma de desarrollo: Java EE
- Arquitectura de componentes software
- Sistema Gades
- Sistema Pegaso

2.3.1. Plataforma de desarrollo: Java EE

El proyecto "*Implementación de una arquitectura de componentes para un sistema de apoyo a la toma de decisiones en Atención Primaria*" **[URI13]** citado anteriormente detalla la elección de la plataforma Java EE de una manera exhaustiva. El presente proyecto utiliza esta plataforma de desarrollo para mantener una homogeneidad entre los diferentes sistemas que cooperan entre sí, por lo que se realizará una descripción de la plataforma para ayudar a comprender el sistema y su funcionamiento, pero no se incidirá en algunos detalles propios de la elección de dicha plataforma al estar expuestos en dicho proyecto.

La plataforma Java EE utiliza la tecnología asociada al lenguaje de programación Java y sus *Interfaces de Programación de Aplicaciones (API)* para la *Edición Empresarial (Enterprise Edition)*. Hay que destacar que esta plataforma es de libre distribución y código abierto por lo que se considera una solución de código abierto.

Java EE permite organizar la estructura del proyecto web entorno a capas, de manera que le otorga una modularidad necesaria si queremos independizar las diferentes funciones para evitar acoplamientos y problemas futuros. Además, esta independencia entre ellas permite la reutilización de componentes para sistemas posteriores.

Las capas utilizadas se dividen en:

- Capa de presentación o de interfaz de usuario.
- Capa de lógica de negocio.
- Capa de datos o del sistema de información de empresa (EIS).

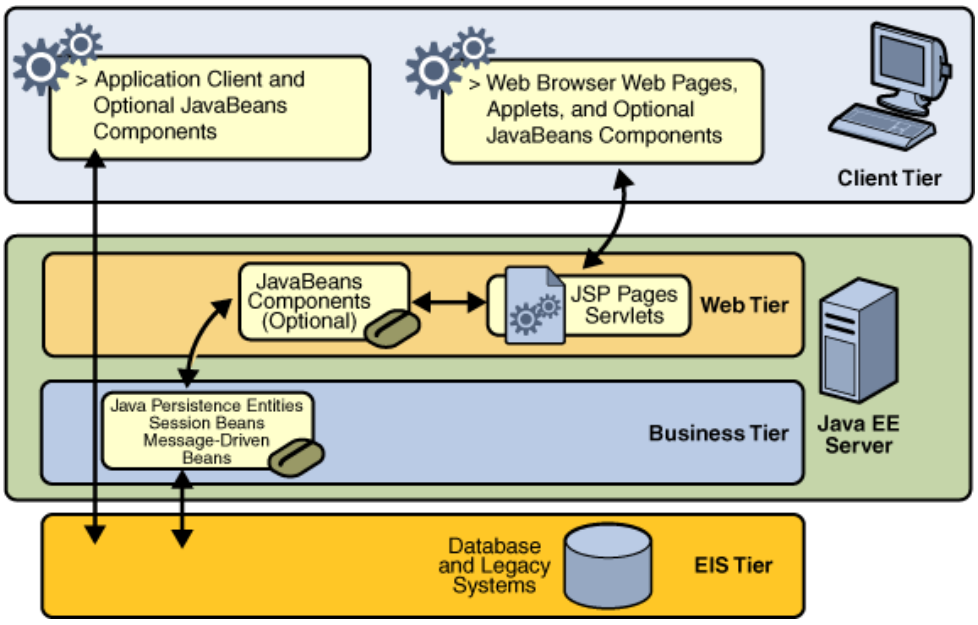


Figura 9: Arquitectura de capas Java EE

Esta estructura permite obtener un sistema modular y reutilizable, refiriéndonos con esto a su división en módulos que realizan funciones diferentes, pero que cooperan entre sí, y a la capacidad que otorgan estos módulos de poder construir otros sistemas mediante el empleo de los mismos.

Este apartado explicará los diferentes componentes que forman cada capa software, sus funciones dentro del sistema y su interacción.

Capa web de presentación

Esta plataforma se apoya en el uso del servicio *World Wide Web* (web) y su protocolo asociado *HyperText Transfer Protocol* (HTTP), de amplia difusión en la actualidad. Las dos plataformas admiten un cliente web, es decir, un browser, navegador o explorador web que actuará de interfaz gráfico con el usuario.

La función de esta capa de presentación es servir de interfaz gráfico para los usuarios de la plataforma. Esta interfaz debe ser capaz de mostrar la información entregada por el sistema, así como de recoger la información que aporte el usuario y las órdenes que realice.

Esta capa de presentación está basada en las tecnologías web, y como tal está formada por una parte cliente y una parte servidora, las cuales intercambian mensajes HTTP conteniendo las peticiones y sus respuestas.

La parte cliente se basa en el uso de los navegadores web como intérpretes del software contenido en las páginas web que constituirán el interfaz que verá el usuario.

Estas páginas web originarán peticiones que serán atendidas por componentes del lado servidor.

Los componentes del lado servidor podrán generar respuestas HTTP en Lenguaje de Marcas de Hipertexto eXtensible (*eXtensible HyperText Markup Language, XHTML*). Estas respuestas son páginas destinadas a ser visualizadas en el cliente web, el navegador.

Los componentes web del lado servidor serán de dos clases diferentes:

- Páginas de Servidor Java (Java Server Pages, JSP)
- Componentes Java web del lado Servidor (Servlets).

A nivel funcional ambos componentes son equivalentes, se puede implementar una interfaz web mediante JSP o Servlet. La diferencia reside en su aspecto formal desde el punto de vista del desarrollo.

JSP es una tecnología que permite a los desarrolladores de software crear páginas web generadas dinámicamente basadas en HTML, XML, u otros tipos de documentos. Su principal función es la construcción de las vistas de usuario, es decir, las páginas que visualizará.

Por su parte, los Servlets son objetos que recibe una solicitud y genera una respuesta basada en esa solicitud, ambas HTTP. El paquete básico de Servlet define los objetos de Java para representar las solicitudes y respuestas de Servlet, así como objetos para reflejar los parámetros de configuración del Servlet y el entorno de ejecución. Al tratarse de clases completamente Java, facilita la ejecución de cierta lógica en los casos en los que sea necesaria para construir la respuesta a una solicitud.

A continuación, en la Figura 10 muestra cómo se realiza el tratamiento de una petición HTTP por parte de una aplicación web Java.

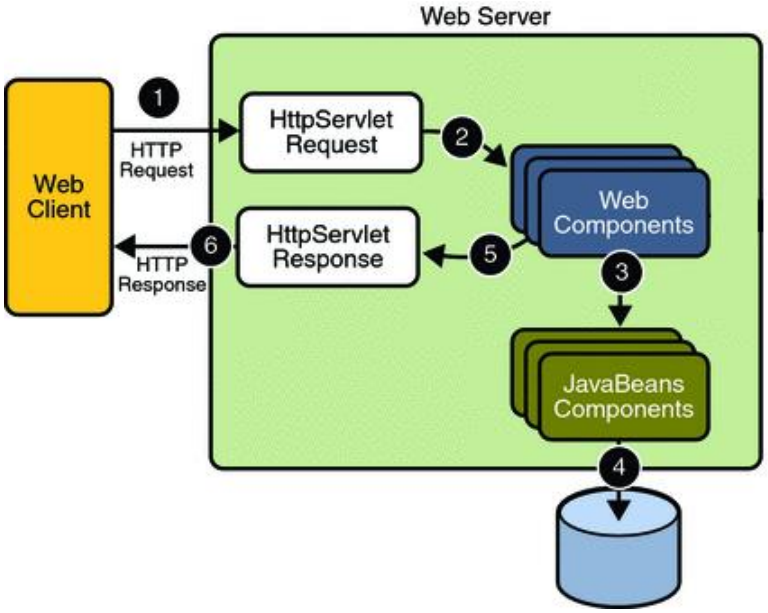


Figura 10: Tratamiento de peticiones y respuestas HTTP en una aplicación web Java

Los pasos que se observan en la figura muestran la interacción entre los componentes de la aplicación. Estos pasos son:

1. El cliente web (navegador, browser) realiza una petición destinada al servidor web de la aplicación.
2. El servidor recibe la petición y la encapsula en un objeto *HttpServletRequest*, el cual contiene toda la información de la petición recibida, y se la entrega a un componente web para su procesamiento.
3. Los componentes web con la información obtenida realizan llamadas a los componentes JavaBeans que contienen la lógica de la aplicación y el acceso a datos.
4. Estos componentes JavaBeans realizan el procesamiento requerido por los componentes web. Realizarán las tareas que les correspondan por sus funciones diferenciándose los encargados del acceso a datos y los encargados de la lógica.
5. El componente web realiza el tratamiento de los datos obtenidos para conformar la respuesta al cliente web. Esta respuesta esta encapsulada en un objeto *HttpServletResponse*.
6. La respuesta HTTP con la información del objeto anterior es enviada por el servidor web hacia el cliente.

Asimismo, para la construcción de la capa de presentación se ha hecho uso de un patrón de diseño: los **ayudantes de vista**. Este patrón pretende simplificar el acceso a la lógica y al acceso a datos para las vistas de usuario.

De este modo, utilizaríamos las páginas JSP para la obtención de las vistas de usuario y los ayudantes de vista estarían representados por Servlets en los que estas vistas se apoyan. Este patrón y su implementación en el sistema se explican con más detalle más adelante en el apartado "Diseño y construcción de la interfaz de usuario".

La *capa de lógica de negocio* consiste en la lógica que realiza las funciones principales de la aplicación: procesamiento de datos, implementación de funciones de negocio, coordinación de varios usuarios y administración de recursos externos como, por ejemplo, bases de datos o sistemas heredados.

Esta capa suele estar formada por componentes firmemente acoplados que se ajustan al modelo de componentes distribuidos de J2EE como, por ejemplo, los objetos Java, los componentes EJB *(Enterprise Java Beans)*. Los servicios de negocios también se pueden crear como servidores independientes como, por ejemplo, un servidor de mensajería o un servidor de calendario empresarial.

Capa web de lógica

La capa de lógica de la aplicación se encarga de realizar el procesamiento asociado a las funciones de negocio. Este procesamiento se implementa utilizando la información obtenida en la capa de presentación y junto con los datos accesibles mediante la capa de datos.

En esta capa, la lógica se encapsula en componentes denominados JavaBeans. Estos componentes son objetos Java que se ajustan a ciertas reglas que permiten aumentar su capacidad de reutilización.

Al situarse entre las capas de presentación y datos mantiene un flujo de información con ambas. Por un lado, **obtiene** información de las capas de presentación y de datos, y por otro lado, **entrega** información a la capa de presentación sobre lo que debe mostrarse y a la capa de datos sobre información que almacenar.

Del mismo modo que ocurría en la capa de presentación, la complejidad del problema hace necesaria la utilización de patrones de diseño cuya finalidad sea la simplificación de las tareas. En concreto, se han utilizado los siguientes patrones:

- **Patrón de delegado de negocio**: su función es reducir el acoplamiento entre capas. Normalmente nos referiremos a los componentes que implementan este patrón como "controladores de lógica".
- **Patrón de manejador de lista de valores**: su utilidad es iterar eficientemente en una lista amplia de datos.
- **Patrón de objeto de transferencia**: su utilidad es encapsular datos a transferir entre capas.

Capa web de acceso a datos

La última capa de este tipo de plataformas es la capa del sistema de información de una organización o *capa de datos*. Esta capa puede estar constituida por algo tan sencillo como un sistema de ficheros o tan complejo como una combinación de sistemas de planificación de recursos empresariales, bases de datos, etc. Siendo lo más habitual, una base de datos en un sistema gestor de base de datos que será independiente del servidor web o del servidor de aplicaciones. En esta capa se proporcionará la persistencia de datos y las operaciones de manejo de dichos datos.

La capa de acceso a datos en una aplicación web Java EE se encarga del procesamiento de la información que se desea almacenar u obtener del sistema de información o base de datos del sistema.

Como se ha descrito anteriormente, la comunicación se realiza mediante objetos de transferencia o colecciones de los mismos.

La capa de acceso está encapsulada en componentes JavaBean que implementan los accesos a la base de datos del sistema. La finalidad de estos componentes es el aislamiento de los accesos a la base de datos con respecto a la lógica y la presentación.

Como se observa a lo largo de esta descripción de la arquitectura del sistema, existirán varios tipos de JavaBeans. Unos pertenecerán a la capa de lógica y contendrán la lógica de negocio; otros pertenecen a la capa de acceso a datos y permiten aislar ese acceso; y otros se utilizarán como objetos de transferencia de datos entre capas.

Para el acceso a datos se ha empleado un patrón cuya función es abstraer y encapsular los accesos a datos, se trata del **patrón de objeto de acceso a datos**. Los objetos que implementen este patrón tendrán en su nomenclatura las siglas DAO, *Data Access Object*.

Estos objetos de acceso utilizarán la interfaz de programación JDBC que permite en manejo de base de datos en Java. Junto con JDBC se utiliza sintaxis SQL para realizar las consultas e inserciones que tendrán lugar en el sistema.

2.3.2. Arquitectura de componentes software de las aplicaciones

Tras describir la plataforma de desarrollo de una manera general, a continuación se procede a realizar una descripción de la arquitectura software de la aplicación web Java en la que se basan Pegaso y Gades.

Partiendo de los requisitos que motivaron las decisiones tomadas:

- El requisito de acceso web al sistema hace necesario el uso de un contenedor web en el lado servidor. Este contenedor que forma parte de la capa web realizará la comunicación mediante protocolo web (HTTP) con la capa cliente que se ejecuta en la máquina cliente.
- La capa cliente estará formada por páginas web interpretadas en un navegador y definidas de acuerdo a los lenguajes asociados a este entorno.
- Serán necesarios interfaces de programación adecuados para acceder al sistema de información constituido por una base de datos relacional y el archivo descriptivo de una ontología.
- No existen aplicaciones o sistemas externos al nuestro que accedan al sistema de información al mismo tiempo, con lo que no existe necesidad de sincronización de las entidades de datos respecto a esos sistemas externos.

Podemos cumplir con todos los requisitos anteriores si decidimos implementar el sistema apoyándonos en un servidor de aplicaciones Java EE.

Cumpliendo con el requisito de usar sistemas de código abierto y software libre es fácil tomar la decisión de utilizar un **servidor web Apache**. Este servidor ofrece suficientes garantías dada su amplia implantación en el mundo web.

Como contenedor web Java utilizaremos la solución que proporciona el mismo proyecto Apache: el **contenedor web Apache Tomcat**.

Respecto a la capa de información o de datos, nuestro sistema utilizará un sistema gestor de base de datos de software libre muy extendido en cuanto a su uso y suficientemente probado en diferentes entornos: el **servidor de base de datos MySQL**.

La Figura 11 muestra la estructura de contenedores y servidores sobre la que se ha construido la solución.

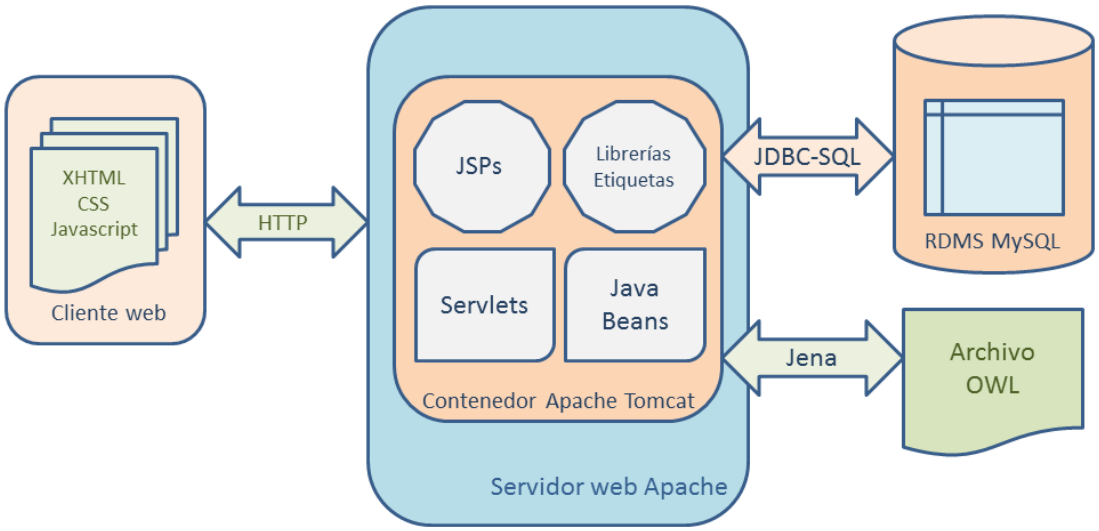


Figura 11. Estructura de contenedores y servidores

Puede observarse como el cliente es un cliente web típico formado por un conjunto de páginas web definidas en los lenguajes asociados al entorno web.

En el lado del servidor web tendremos dentro del contenedor Tomcat diferentes tipos de componentes de software Java:

- Las páginas *JSP*, encargadas de generar todo el código de la parte cliente de forma dinámica desde el servidor.
- Las librerías de *etiquetas personalizadas* que se desarrollarán para ser usadas dentro de las páginas JSP.
- Los *Servlets* que actúan como intermediarios entre JSP y la capa de lógica, siendo los encargados de procesar las peticiones y respuestas HTTP.
- Los objetos *Java Bean*.

Estos objetos Java Bean estarán clasificados en dos tipos diferentes:

- Java Beans de lógica de negocio, en los que se implementará la lógica de la aplicación. Los denominaremos también controladores de lógica.
- Java Beans de acceso a datos, que contendrán el código de acceso a datos. Los denominaremos también objetos de acceso a datos (DAO).

El acceso a datos se realizará mediante el API Java DataBase Connectivity (JDBC) que permite el acceso a los datos mediante el uso de sentencias del Structured Query Language (SQL) embebido dentro del código Java.

El acceso a los datos de la ontología contenidos en el archivo OWL (Ontology Web Language) se realizará mediante el API Apache Jena. Estos elementos son explotados en el presente proyecto y, por lo tanto, se detallarán más adelante.

2.3.3. Sistema Pegaso

El primero de los sistemas previos es Pegaso. Este sistema atiende a la necesidad del personal médico de atención primaria de disponer de herramientas específicas para la ayuda en la detección de trastornos neurológicos en niños, en este caso, trastornos del lenguaje.

La arquitectura funcional del sistema Pegaso tiene como objetivo de facilitar la interacción entre los diferentes actores implicados, las plataformas distribuidas de gestión fiable de la información, los modelos de razonamiento y los procesos de actuación acordes con el modelo sanitario en el que se ubica. La siguiente figura resume esta interacción.

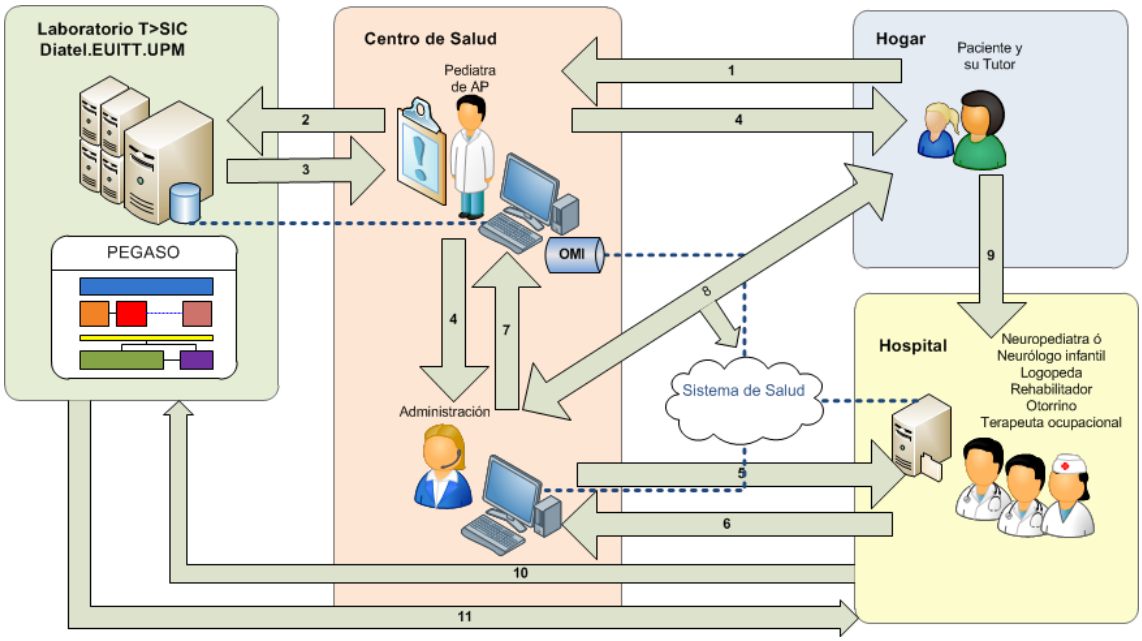


Figura 12: Arquitectura funcional del sistema Pegaso

A continuación se hará una breve descripción de en qué consiste cada uno de los pasos que recoge la Figura 12:

Paso 1. El niño acude al pediatra de familia acompañado de un miembro de su familia.

Paso 2. El pediatra de Atención Primaria decide utilizar Pegaso para evaluar si existe algún trastorno del lenguaje en el niño, en cuyo caso se realizará la derivación precoz al especialista correspondiente o bien se adelantará la próxima visita del niño con objeto de realizar una nueva evaluación. El pediatra interactúa con el sistema realizando la introducción de información correspondiente.

Paso 3. El sistema devuelve el resultado al pediatra.

Paso 4. Obtenida la respuesta, existen dos posibilidades:

a) El resultado es que todo es normal en cuyo caso el niño no modifica el calendario de sus visitas al pediatra.

b) El resultado modifica el calendario de visitas al pediatra, o se indica la derivación al especialista pertinente.

Paso 5. Se realiza la petición de cita con el especialista correspondiente del hospital.

Paso 6. Se recibe respuesta a la petición de cita con el hospital,

Paso 7. Los datos de la cita con el especialista son recibidos por el pediatra.

Paso 8. Los datos de la cita llegan al niño y a su familia.

Paso 9. El niño acude a la cita concertada con el especialista.

Paso 10. El especialista consulta la evaluación realizada para el caso de estudio correspondiente.

Paso 11. El sistema devuelve el resultado al especialista.

Casos de uso de Pegaso

A continuación, se muestra el diagrama de casos de uso de Pegaso.

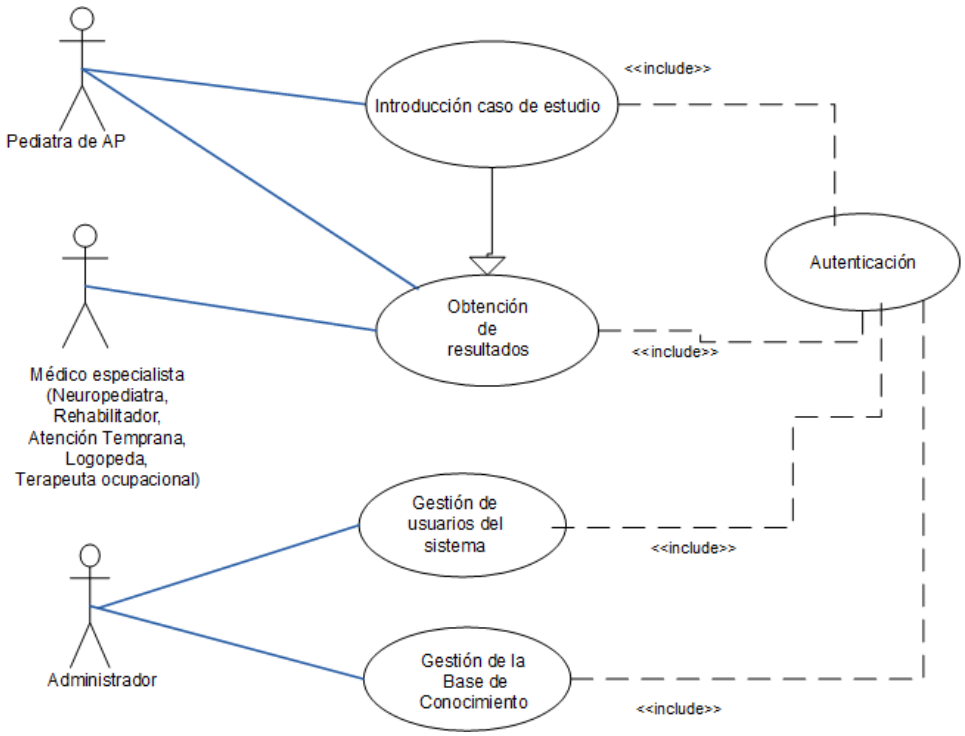


Figura 13: Caso de uso de Pegaso

La Figura 13 resume las funcionalidades comentadas anteriormente mediante la arquitectura funcional del sistema. De una manera concisa se observa la participación de los distintos actores en tareas específicas:

- Los pediatras introducen los casos de estudio, es decir, realizan las evaluaciones y dan de alta en el sistema a los niños. También pueden consultar los resultados de las evaluaciones anteriores.
- Los médicos especialistas tienen acceso a los resultados, de manera que puedan conocer los datos obtenidos antes de ser derivado el paciente.
- El administrador tiene acceso al sistema para realizar el mantenimiento de los usuarios (altas, bajas o modificaciones) y de la base de conocimiento.

Todas las funcionalidades del sistema están accesibles previa autenticación del usuario.

2.3.4. Sistema Gades

Gades surge de la posibilidad observada de crear un sistema similar a Pegaso que sea utilizado por los **educadores**, personal más cercano a los niños y que pueden realizar una observación más completa del comportamiento y desarrollo del niño.

La arquitectura funcional del sistema resultante ha de facilitar la interacción dinámica entre los actores implicados, las plataformas distribuidas de gestión fiable de la información, los modelos de razonamiento y los procesos de actuación acordes con el modelo sanitario en el que se ubica. La Figura 14 resume esta interacción que se explica con mayor detalle en los apartados siguientes:

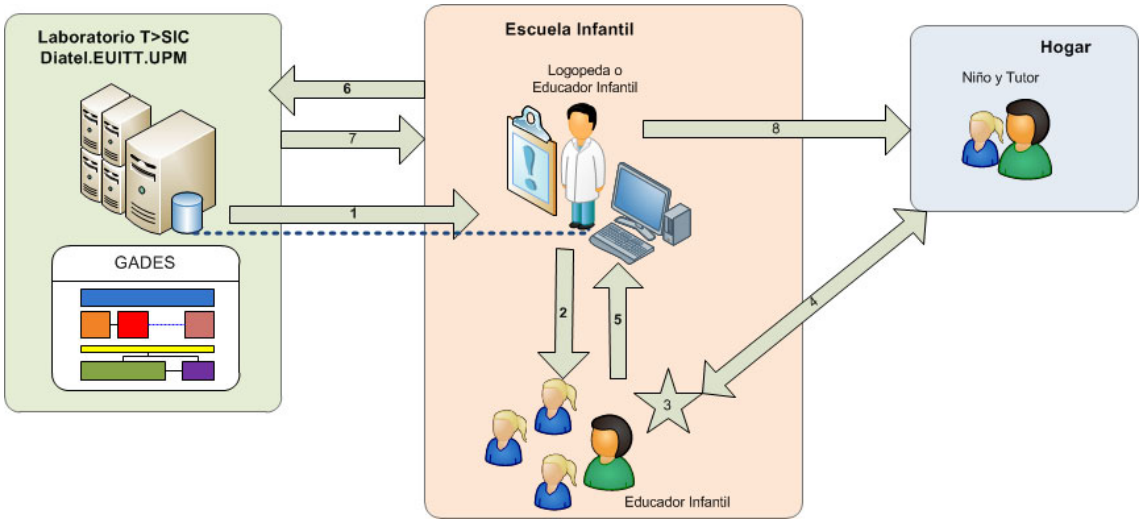


Figura 14: Arquitectura funcional del sistema Gades

A continuación, se hará una breve descripción de las acciones a realizar en cada uno de los pasos que recoge la Figura 14:

Paso 1. El usuario del sistema Gades (educador infantil o logopeda del colegio), realiza la evaluación del lenguaje de sus alumnos utilizando Gades. Para ello, obtendrá en formato papel un cuestionario con los hitos de desarrollo que están almacenados en la base de conocimiento y que debe observar según la edad del niño en meses.

Paso 2. El usuario le proporciona al educador infantil el cuestionario con la observación que debe realizar para cada niño

Paso 3. El educador infantil realizará la observación del nivel de adquisición del lenguaje de sus alumnos. Esta observación durará un tiempo variable entre una y dos semanas dependiendo del tiempo que necesite para responder a las preguntas.

Paso 4. Puede que para responder a alguna de las preguntas se deba recurrir a los padres, ya que el niño puede comportarse de forma distinta en el colegio y en casa.

Paso 5. Cuando el educador finaliza todos los cuestionarios de sus alumnos, se los devuelve al usuario encargado de realizar las evaluaciones del lenguaje utilizando Gades.

Paso 6 y 7. El usuario interactúa con el sistema Gades realizando el proceso de evaluación del lenguaje de todos los niños que participan en el estudio.

Paso 8. Cuando el resultado obtenido para algún alumno sea alarmante el centro educativo valorará la opción de informar a los padres de manera que se pueda realizar un diagnóstico completo del posible trastorno detectado por Gades.

Casos de uso de Gades

Del mismo modo, la Figura 15 resume las funcionalidades comentadas anteriormente mediante la arquitectura funcional del sistema, en ella se puede observar los casos de uso que cubre el sistema Gades.

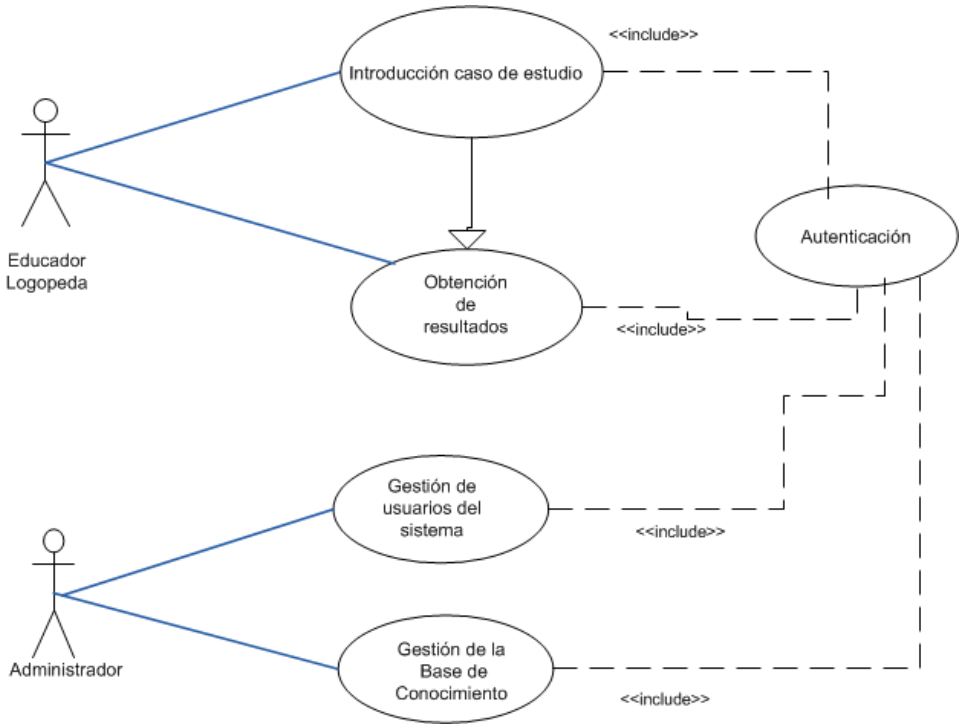


Figura 15. Casos de uso de Gades

Los actores que interaccionan con el sistema son:

- El educador infantil o el logopeda, los cuales tienen el papel del usuario que interactúa con Gades directamente, por ejemplo, realizando las evaluaciones del lenguaje.
- El administrador tiene acceso al sistema para realizar el mantenimiento de los usuarios (altas, bajas o modificaciones) y de la base de conocimiento, encargándose de realizar modificaciones en la ontología.

Se tiene previsto que tras la puesta en producción del Sistema de Aprendizaje Colaborativo Supervisado, la gestión de la evolución de la base de conocimiento no dependa de este usuario administrador, sino que quede en manos de los usuarios del sistema y de un grupo de usuarios expertos, encargados respectivamente de proponer y aceptar cambios en la estructura de la base de conocimiento.

Capítulo 3. Descripción de la solución propuesta

La solución desarrollada a lo largo de este proyecto se enmarca dentro del contexto descrito en apartados anteriores. La existencia de un sistema previo supone la adaptación de este sistema a decisiones ya tomadas en cuanto a diseño e implementación.

Por lo tanto, este proyecto podría considerarse una evolución dentro de una plataforma de mayor envergadura, conformada por Pegaso, Gades y él mismo (sistema **Galatea**). Del mismo modo se trata de una plataforma independiente descrita por los objetivos enumerados en la introducción de la presente memoria.

Este proyecto asume como válidas las decisiones de diseño tomadas en los sistemas ya desarrollados.

En los apartados siguientes se describen detalladamente la solución propuesta por este proyecto. Esta descripción estará guiada por el siguiente recorrido:

- Plataforma tecnológica.
- Análisis del sistema Galatea
- Construcción de la Base de Datos.
- Diseño y construcción del acceso a datos.
- Diseño y construcción de la lógica funcional o de negocio.
- Diseño y construcción de la interfaz de usuario.
- Gestión de errores de la aplicación.
- Archivos de configuración de la aplicación.

3.1. Plataforma tecnológica

Como se ha explicado anteriormente, la elección de la plataforma de desarrollo para este proyecto viene dada por la coherencia del mismo con respecto a las plataformas existentes y, por tanto, la utilización de las mismas tecnologías de desarrollo, así como la aceptación de la arquitectura y métodos de diseño ya utilizados.

Las particularidades del sistema hacen necesaria la explotación de las tecnologías utilizadas en Pegaso y Gades. Aunque en este proyecto no se aplican del mismo modo.

3.1.1. OWL: Ontology Web Language y Jena API

OWL es una familia de lenguajes de representación de conocimiento o lenguajes de ontologías para la creación de ontologías o bases de conocimiento. Estos lenguajes están caracterizados por una semántica formal y la utilización de los lenguajes RDF/XML para la Web Semántica. OWL está aprobado por el W3C (World Wide Web Consortium) y ha despertado el interés académico, médico y comercial.

Tanto OWL como Jena están relacionadas con el tratamiento de la ontología por parte del sistema. Por un lado, el lenguaje OWL está siendo utilizado para caracterizar la ontología y generar el fichero OWL (con la base de conocimiento) que será utilizado por Pegaso y Gades. Por otro lado, la API Jena, la cual se emplea para el manejo mediante código de dicha ontología y fichero OWL; resulta imprescindible para la creación de la base de datos del sistema Galatea como veremos más adelante.

El conocimiento exhaustivo del lenguaje OWL no es objeto del presente proyecto, por ello se describe la estructura básica del mismo con el fin de comprender las tecnologías utilizadas, así como, el porqué de los pasos seguidos.

Ahora bien, este lenguaje es el utilizado para caracterizar la base de conocimiento de nuestro sistema empleando para ello una notación específica. Esta notación utiliza una determinada terminología, formada por [WIK14]:

- **Instancias**. Una instancia es un objeto. Corresponde a la descripción lógica de un individuo.
- **Clases**. Una clase es una colección de objetos y se corresponde con la descripción lógica de un concepto. Las principales características son:
    - Una clase puede contener individuos, las instancias de la clase.
    - Una clase puede tener cualquier número de instancias.
    - Una instancia puede pertenecer a ninguna, una o más clases.

Del mismo modo, una clase puede ser una subclase de otra, heredando las características de su superclase padre. Por ejemplo, la clase Empleado podría ser superclase de Gerente o Trabajador.

- **Propiedades**. Una propiedad es una relación binaria dirigida que especifica las características de clase. Son atributos de las instancias y, a veces, actúan como valores o enlaces a otras instancias. Las propiedades pueden tener dominio y rango.
    - **Propiedades de tipos de datos**. Las propiedades de tipos de datos son las relaciones entre las instancias de clases y las etiquetas RDF o los tipos de datos de XML.
    - **Propiedades de objeto**. Las propiedades del objeto son las relaciones entre instancias de dos clases. Por ejemplo, *propiedad de* puede ser una propiedad de objeto de la clase Vehículo y puede tener un rango que es la clase Persona.

Por su parte, el API de **Jena** es un entorno Java gratuito y de libre acceso para la creación de la Web Semántica y aplicaciones de *Linked Data*. Proporciona unas amplias bibliotecas de Java para ayudar a desarrollar código que maneje RDF, OWL y SPARQL en línea con las recomendaciones del W3C publicadas. Jena incluye un motor de inferencia basado en reglas para realizar el razonamiento basado en ontologías OWL y RDFS, y una variedad de estrategias de almacenamiento para almacenar tripletas RDF en la memoria o en el disco.

Jena permite el manejo de modelos OWL, siendo capaz de obtener una representación de la ontología mediante objetos Java que podemos utilizar en nuestro programa. Lo mismo ocurre con clases, subclases y restricciones, en lo cual está basado principalmente nuestro fichero OWL.

Estas facilidades hacen posible la obtención del conocimiento a partir del fichero OWL, obtención que se detalla junto con la base de datos del sistema.

Por su parte, la elección de la API Jena es producto de un estudio de mayor envergadura realizado por Óscar Palomo en su proyecto: "*Análisis de datos y modelado simbólico para aplicación de detección temprana de trastornos del lenguaje*" **[PAL14]**. En él, se describen las posibles API contempladas para el procesado de los ficheros OWL desde código, tanto para la obtención del conocimiento como en este caso, como para la creación automática de un fichero OWL desde la información contenida en la base de datos.

Este proyecto describe las siguientes APIs:

- Apache Jena.
  Aúna principalmente sus funcionalidades entorno a: RDF API, que permite acceder, manipular y escribir información almacenada en gráficos RDF; y Ontology API, permite trabajar con modelos (*Model* es el contenedor principal de la información RDF), RDF Schema y OWL para añadir semántica adicional a los datos RDF.
- OWL API.
  Es un API para Java, de software libre y de código abierto, que permite crear, manipular y serializar[1] ontologías OWL. La última versión del API está enfocada hacia OWL 2
- Protégé-OWL.
  Es un API para Java, software libre y de código abierto, para los lenguajes OWL y RDF/RDF Schema. Este API provee clases y métodos para cargar y guardar archivos OWL, para consultar y manipular modelos de datos de OWL, y para llevar a cabo razonamientos basados en razonadores DL. Además, este API está optimizado para la implementación de interfaces gráficas de usuario.

Tras la descripción de las mismas, se concluye la elección de **Jena** debido a que ya se usaba en las aplicaciones Gades y Pegaso con anterioridad y, producto de esta utilización, existía una familiarización que no había con las otras API.

3.2. Análisis del sistema Galatea

Una vez definida la plataforma a utilizar y su arquitectura, procedemos a realizar el análisis del sistema.

Comenzamos el análisis del sistema situando éste en su contexto funcional, es decir, en colaboración con los sistemas Pegaso y Gades mencionados. La Figura 16 muestra esta arquitectura funcional, recogiendo como interaccionan los distintos

[1] La **serialización** es el proceso de codificación de un objeto en un medio de almacenamiento (como puede ser un archivo, o un buffer de memoria) [http://es.wikipedia.org/wiki/Serialización]

usuarios con los sistemas Gades, Pegaso y Galatea, así como el orden en el que debe producirse dicha interacción.

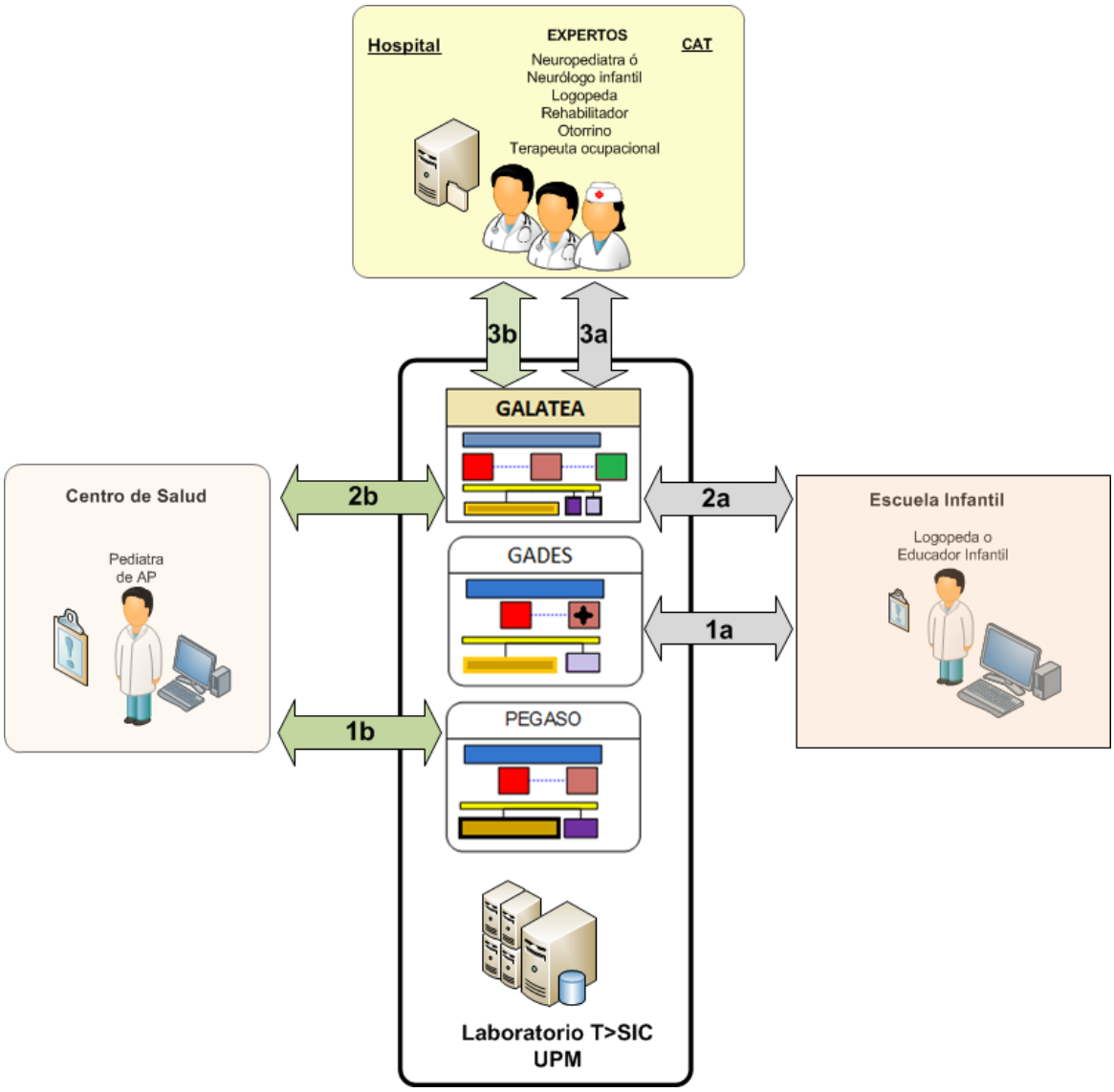


Figura 16: Arquitectura funcional del sistema Galatea

Como se observa en la figura existen tres acciones diferenciadas:

1. **Interacción con Gades (1a) y Pegaso (1b)** por parte de sus respectivos usuarios. En este paso, los usuarios se conectan a las plataformas para realizar las evaluaciones del lenguaje, por un lado los educadores y logopedas y por otro el personal médico (pediatra).
2. **Recolección de propuestas con Galatea (2a y 2b)**. Los usuarios de Gades y Pegaso utilizan **Galatea** para realizar las propuestas de cambios sobre la base de conocimiento de ambas plataformas. La solución planteada en el presente proyecto resuelve las necesidades planteadas en este paso.

3. **Evaluación de las propuestas por un grupo de expertos (3a y 3b).** Los expertos utilizarán las propuestas recogidas para decidir, de una manera consensuada y colaborativa, los cambios que deben llevarse a cabo en la base de conocimiento de cada una de las plataformas. Esta tarea queda fuera del alcance de este Proyecto Fin de Grado.

A continuación, en este apartado se explican las funcionalidades del sistema con el objetivo de profundizar en las funciones y uso del sistema como paso previo a la descripción de las diferentes partes y su coordinación.

Esta descripción se va a realizar utilizando el Lenguaje Unificado de Modelado (Unified Modeling Language, UML), el cual permite modelar y describir sistemas software mediante la utilización de diagramas.

A lo largo de este apartado vamos a utilizar varios tipos de diagramas. El primero de ellos es el diagrama de **casos de uso,** que realiza una descripción de las actividades que pueden ser realizadas en el sistema. Quienes realizan esas acciones se denominan actores, los cuales serán educadores infantiles y logopedas como se indicaba en los requisitos del sistema.

Por otro lado, para describir los algoritmos que se deben seguir para realizar determinadas acciones se utilizarán diagramas de **actividad.** Estos diagramas permiten observar el flujo de trabajo y las acciones realizadas para completar un caso de uso concreto.

3.2.1. Requisitos funcionales del sistema Galatea

Las particularidades del sistema Galatea influyen en cierta manera en algunas de las decisiones tomadas. Por todo ello, a continuación se analizarán los factores y requisitos del sistema.

El primer paso a realizar antes de tomar cualquier decisión sobre el sistema es analizar los diferentes factores que le afectan. Estos factores conforman los requisitos de nuestro sistema, ya que deben ser cumplidos con la elección de la plataforma y su diseño.

Los principales factores que debemos tener en cuenta afectan: a las funcionalidades a ofrecer y a los usuarios de la plataforma. Por lo tanto, los **requisitos funcionales** del sistema estarían constituidos por:

- El sistema solicitará que el usuario se identifique mediante usuario y contraseña.
- El usuario podrá realizar propuestas sobre preguntas de la BC existentes o nuevas.
- El usuario podrá consultar las propuestas realizadas anteriormente.
- Las propuestas serán almacenadas con el nombre del usuario que las realizó, de manera que puedan asociarse a él en las búsquedas.

Se deben observar también los requisitos que no afectan directamente a las funcionalidades, pero que suponen limitaciones a tener en cuenta y que detallamos como **requisitos no funcionales**:

- La autenticación en el sistema será necesaria para poder realizar propuestas o consultas.
- El acceso a la plataforma se realizará mediante una interfaz web.
- El sistema debe manejar una base de datos actualizada que represente la información ya existente en la base de conocimiento de los sistemas Gades y Pegaso.
- El sistema Galatea extraerá, de forma automática, el conocimiento existente en el fichero que almacena la ontología.
- El sistema tiene que aplicarse de forma concreta y limitada en el tiempo, de forma que no suponga una carga de trabajo adicional para el usuario del sistema.
- El sistema recogerá la información necesaria para poder formular correctamente las propuestas.

La siguiente figura recoge los pasos que deben realizarse para conseguir la evolución supervisada (por un grupo de expertos) de la información recogida en el archivo OWL de Gades/Pegaso (archivo con la información de la Base de Conocimiento).

A continuación se describen las tareas a realizar en cada uno de los pasos que recoge la Figura 17. Están dentro del alcance de este Proyecto Fin de Grado, las tareas 1, 2, 3.

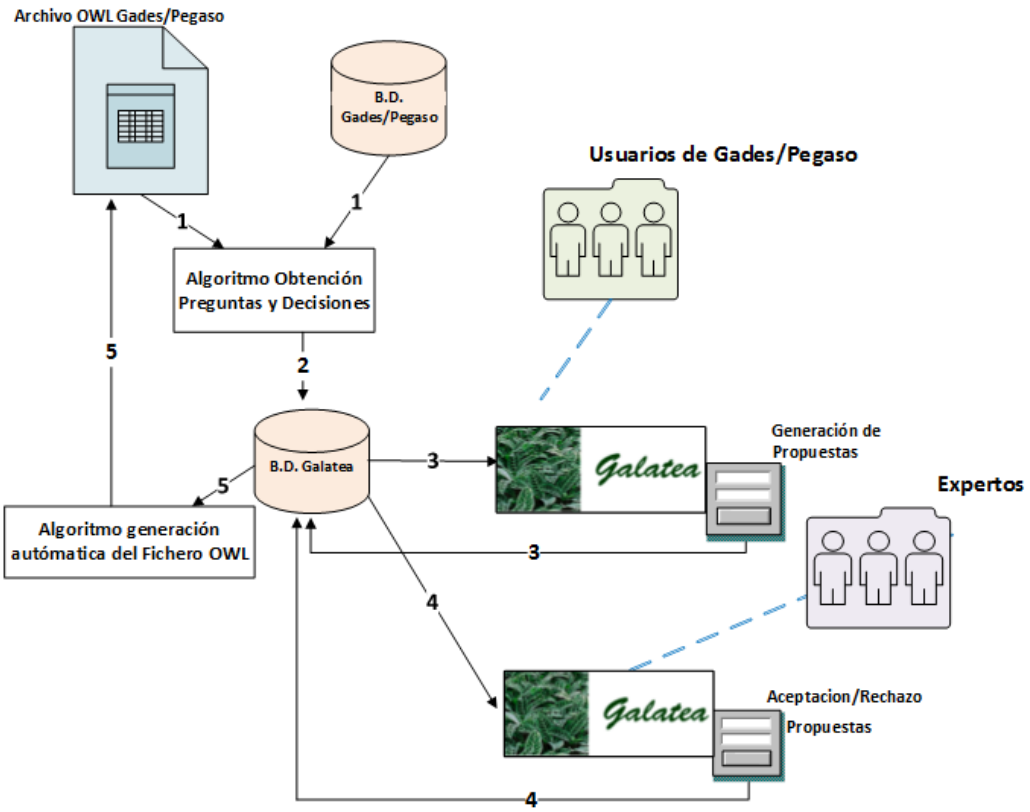


Figura 17: Diagrama de interacción Galatea

Alcance del PFG

Pasos 1 y 2. A partir del archivo con la base de conocimiento (OWL) y la información sobre la descripción de las preguntas (recogida en la BD de Gades/Pegaso), se obtiene la información inicial de la BD de Galatea. Tras este paso se rellenan las tablas de la BD de Galatea con las preguntas y decisiones que están almacenadas inicialmente en el archivo OWL.

Paso 3. Los usuarios de Gades/Pegaso, utilizando la plataforma Galatea, pueden introducir las propuestas de cambio (de preguntas existentes) o propuestas de nuevas preguntas, sobre la información recogida en la base de conocimiento.

Paso 4. El grupo de Expertos de Gades/Pegaso, utilizando la plataforma Galatea, pueden aceptar o rechazar las propuestas de cambio que fueron introducidas en el paso 3.

Paso 5. Con las propuestas que hayan sido aceptadas, se genera automáticamente el fichero OWL, con la nueva versión de la Base de Conocimiento.

Teniendo en cuenta todos estos factores o requisitos, así como la existencia de las plataformas Pegaso y Gades y su implementación, se ha elegido **Java Edición para Empresa (Java Platform Enterprise Edition)**, en adelante Java EE, como plataforma de desarrollo.

A continuación, se describen las características de esta plataforma y cómo se adaptan al sistema requerido.

De una manera general, la Figura 18 ilustra los diferentes casos de uso que encontramos en el sistema.

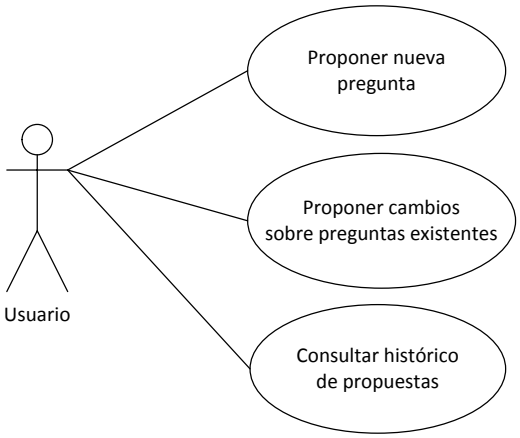


Figura 18: Diagrama de casos de uso del sistema

Por el tipo de operaciones que se realizan en cada uno de ellos, podemos agrupar los casos de uso en dos grupos de funcionalidades. Comenzaremos describiendo las funcionalidades relacionadas con las propuestas que los usuarios de la plataforma pueden realizar.

3.2.2. Funcionalidades asociadas a la realización de propuestas

Estas funcionalidades están directamente relacionadas con el objetivo de conseguir un sistema evolutivo, ya que permiten recoger una realimentación muy valiosa de los usuarios de los sistemas Pegaso y Gades.

Estas funcionalidades están organizadas en dos ámbitos:

- Propuesta de nuevas preguntas.
- Propuesta de cambios sobre preguntas existentes.

En cualquiera de los casos los actores que participan son los mismos: los usuarios de Pegaso y Gades, es decir, pediatras, logopedas, educadores, etc.

En el caso de la **propuesta de una nueva pregunta**, la actividad está compuesta principalmente por elementos para la captura de datos, ya que lo que se busca es introducir una nueva entrada en la base de conocimiento de la plataforma.

Asimismo, esta captura de datos general incluye una serie de actividades más reducidas:

- Introducción de la descripción de la pregunta, la cual debe ser clara, concisa y no contener ambigüedades. Un error en la descripción puede causar que no sea válida.
- Selección de año y mes. Este campo representa la edad en meses en la que se realizará la pregunta y, por lo tanto, en la que el niño debe haber superado el hito indicado en la pregunta.
- Selección/introducción de las decisiones del sistema. Este paso incluye la posibilidad de seleccionar una o varias de las decisiones existentes en el sistema. También existe la posibilidad de introducir manualmente una "propuesta de decisión del sistema", la cual sería posteriormente validada del mismo modo que las propuestas de pregunta.
- Introducción de comentarios. Estos comentarios sirven para justificar la propuesta y forman parte de los conocimientos que los expertos tendrán en cuenta posteriormente.

Para ilustrar de una manera ordenada estas actividades y el comportamiento del sistema, se muestra la Figura 19.

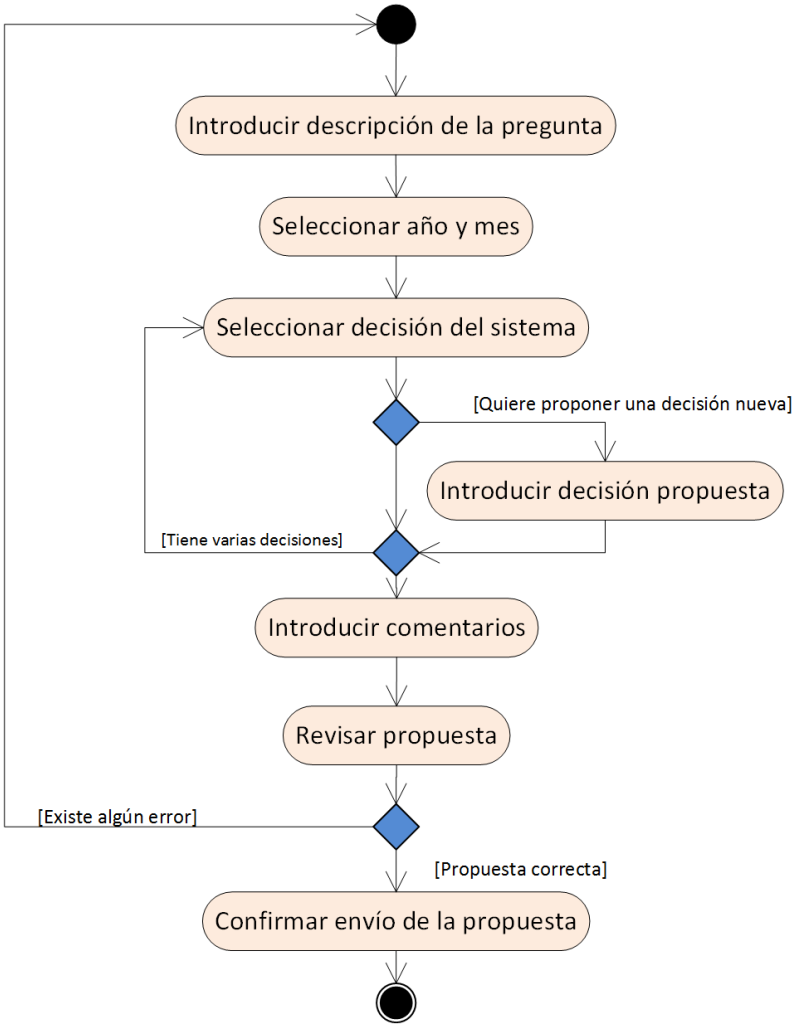


Figura 19: Diagrama de actividad "Proponer nueva pregunta"

Por otro lado, la **propuesta de cambios sobre preguntas existentes**. Esta función del sistema ofrece la posibilidad de consultar las preguntas de la base de conocimiento para realizar propuestas de cambio sobre ellas. Del mismo modo que anteriormente, esta función está formada por otras de menor tamaño:

- Selección de las preguntas que se quiere editar. Se realiza mediante la búsqueda por año, después de lo cual se mostrarán los meses con preguntas y el número de preguntas que tiene. Se eligen uno o varios meses de los cuales se mostrarán sus preguntas, posteriormente se seleccionaran las preguntas concretas (una o varias) que se van a editar.
- Edición de cada una de las preguntas seleccionadas. Cada propuesta se inicia seleccionando el motivo del cambio. Tras la elección se muestran los campos requeridos en función del motivo. Siempre se

solicitarán comentarios que, como en la propuesta de nuevas preguntas, sirvan a los expertos para juzgar la propuesta.

- Revisión y confirmación de las propuestas una vez realizadas. Mediante una vista de resumen de todas las propuestas, se permite revisar las propuestas realizadas y volver a editarlas si se encuentra algún error, o confirmarlas y realizar el **envío definitivo** para su almacenamiento. Es importante informar al usuario, en este momento, de que el envío de propuestas de cambio es definitivo, y que no se van a poder introducir cambios a las propuestas.

La Figura 20 ilustra el recorrido de la tarea, describiendo el flujo de trabajo que se realiza.

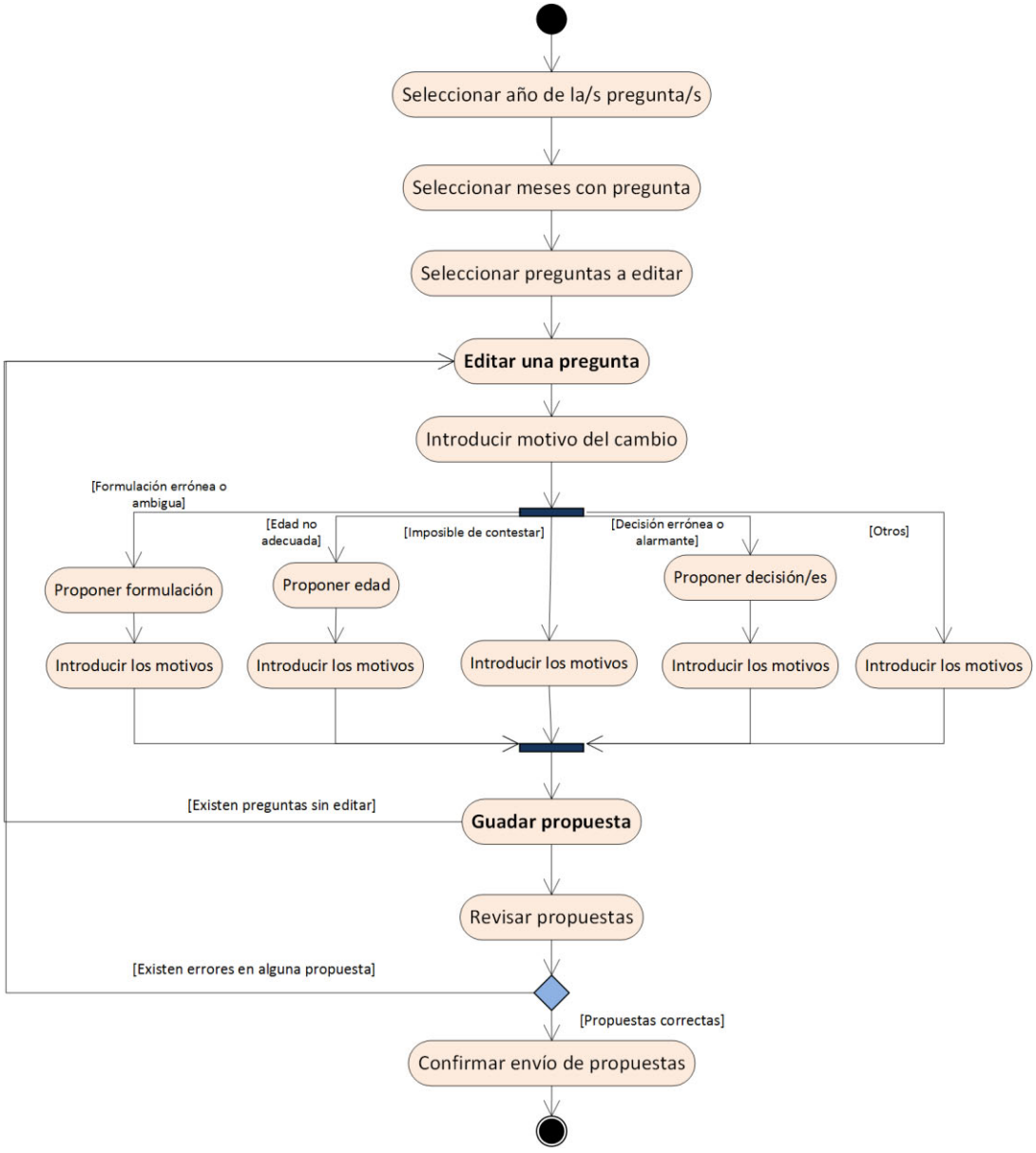


Figura 20: Diagrama de actividad "Proponer cambios en preguntas existentes"

3.2.3. Funcionalidad de consulta de propuestas

Esta función permite al usuario de la plataforma consultar las propuestas que ha realizado. En el proceso de consulta no se permiten realizar actualizaciones de información. La consulta puede realizarse en base en dos posibles criterios:

- Las propuestas realizadas durante la sesión actual, que al tratarse de una interfaz web el término sesión se refiere al período de tiempo entre la autenticación del usuario y su salida del sistema;
- Las propuestas realizadas por ese usuario en cualquier momento, facilitando al usuario la consulta de todas las propuestas introducidas en Galatea.

3.3. Construcción de la Base de Datos

Como soporte fundamental del sistema de información de nuestra plataforma disponemos de una base de datos dentro del *Sistema Gestor de Base de Datos Relacional MySQL*.

La función fundamental de la base de datos será almacenar persistentemente toda la información manejada por la aplicación desarrollada.

Por su parte, el sistema gestor de la base de datos proporcionará las herramientas o mecanismos para añadir, borrar, modificar y consultar los datos.

Esta base de datos parte del cumplimiento de dos necesidades del sistema: la representación de la información recogida en la base de conocimiento (de Gades y Pegaso) y el almacenamiento de las propuestas de cambio sobre la misma. Debido a la independencia entre ambas tareas las trataremos por separado, comenzando por la creación de la representación de la base de conocimiento y continuando con la base de datos que maneja las propuestas.

3.3.1. Base de datos que representa la información de la base de conocimiento de Gades y Pegaso

Como se indica en los objetivos del presente proyecto, el sistema tiene la necesidad de crear automáticamente una base de datos que represente la base de conocimiento de los sistemas Pegaso y Gades. Esta base de datos debe, cumpliendo con los requisitos, mantener una información actualizada ya que el objetivo final del conjunto de Pegaso, Gades y Galatea es la evolución automática de esa base de conocimiento, por lo que una representación estática de la misma sería ineficiente e inútil.

Por otro lado, la base de conocimiento está contenida en un fichero OWL, tecnología comentada en el apartado anterior, del cual debemos obtener la información necesaria para nuestra base de datos relacional.

Para obtener la información contenida en el fichero OWL se ha desarrollado un programa que permite recoger las clases que componen la ontología y las restricciones que establecen sus relaciones. El algoritmo desarrollado viene definido por la estructura interna de la ontología, así como por las particularidades de la API utilizada, Jena. La estructura interna puede observarse en la Figura 21, donde cada punto corresponde a una clase y éstas siguen una distribución en árbol.

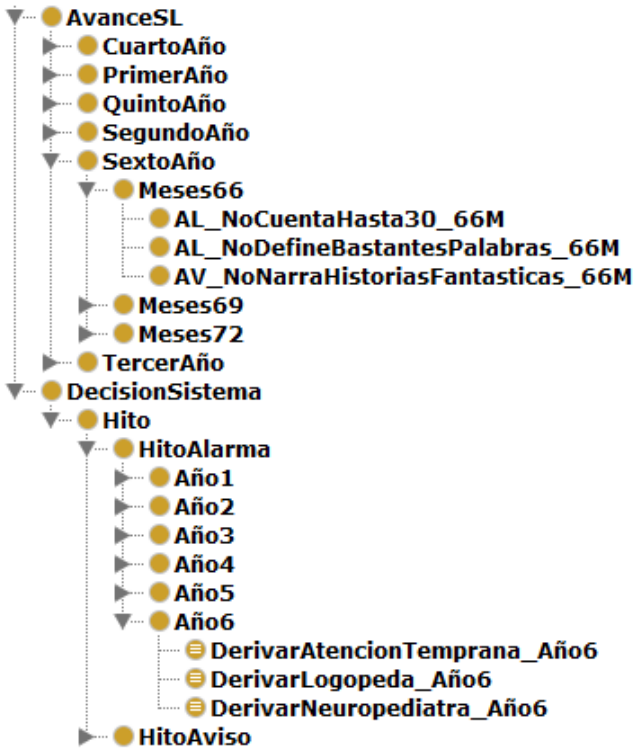


Figura 21: Estructura del fichero OWL (Protégé)

Por lo tanto, el algoritmo sigue esta distribución de árbol y la utiliza para recoger toda la información contenida en el fichero OWL. Dicho algoritmo se apoya en objetos Java específicos que representan lo que posteriormente será **la base de datos de Galatea**. Estos objetos son:

- **Pregunta**: representa una de las preguntas que realizan Pegaso o Gades en las evaluaciones.
- **Decisión**: representa una decisión del sistema.
- **Relación**: contiene las preguntas que, de no responderse correctamente, produciría una determinada decisión.

Estos objetos formarán las tablas de la base de datos, las cuales se detallan más adelante. A continuación, se muestra el algoritmo utilizado, como programa principal, para la obtención de: las preguntas, decisiones y relaciones entre decisiones y preguntas, desde el fichero OWL. El programa principal utiliza objetos de las clases Pregunta, Decision y Relación.

Algoritmo para la obtención de Preguntas (individuo y forma abreviada) y Decisiones

1. **Obtener las clases AvanceSL y DecisionSistema**, que son las clases padre de los árboles que forman las preguntas y las decisiones del sistema.
2. **Iterar sobre la clase AvanceSL** para obtener las preguntas:
   - Mientras haya **años** que procesar:
     - Mientras haya **meses** en el año que procesar:
       Obtener el **número** del mes.
       Mientras haya **preguntas**:
       Obtener **descripción** (abreviada) de la pregunta.
       Detectar si se trata de una **alarma o aviso**.
       Obtener el **individuo** de la pregunta y su descripción.
       Añadir el objeto Pregunta a la colección.
       Fin Mientras.
     - Fin Mientras.
   - Fin Mientras.
3. **Iterar sobre la clase DecisionSistema** para obtener las decisiones del sistema.
   - Para cada tipo de decisión (HitoAviso e HitoAlarma):
     - **Obtener el tipo de decisión** (alarma o aviso).
     - Mientras haya **años** que procesar:
       Obtener año de la decisión.
       Obtener descripción de la decisión.
       Añadir el objeto Decisión a la colección.
       Obtener relaciones con las preguntas (esta acción se realiza mediante el método getRelaciones(), que se explicará a continuación).
     - Fin Mientras.
   - Fin Para.
4. **Almacenar los objetos Pregunta, Relación y DecisionSistema** en la base de datos de **Galatea**.

Con el algoritmo anterior, podemos obtener los meses de cada uno de los hitos observando la clase Mes a la que pertenece; si se trata de un aviso o de una alarma, ya que el inicio de las clases de *AvanceSL* así lo indica mediante "AL" para las alarmas y

"AV" para los avisos; y una versión abreviada de las preguntas, que es la descripción de las clases que componen las preguntas (Figura 21: "AL_NoCuentaHasta30_66M").

Del mismo modo, obtenemos las decisiones del sistema para cada año y mes a partir de la clase *DecisionSistema*, distinguiendo los avisos y alarmas mediante la clase padre a la que pertenecen (HitoAlarma e HitoAviso).

Ahora bien, la parte más importante del conocimiento contenido en el fichero OWL está en las relaciones entre las preguntas y las decisiones del sistema, es decir, la correspondencia entre las decisiones del sistema y las preguntas que pueden provocar dichas decisiones. Esto se realiza mediante un método específico, el cual se detalla a continuación:

Algoritmo para la obtención de relaciones entre una decisión y sus preguntas

MÉTODO	getRelaciones()
PARÁMETROS DE ENTRADA:	**posHito** – Identificador de la decisión (de la que se quieren obtener las preguntas relacionadas). **clase** – Clase que representa a la decisión. **listaPreguntas** – Colección que almacena los objetos Pregunta obtenidos anteriormente.
PARÁMETROS DE SALIDA:	**listaRelaciones** – Colección que almacena los objetos Relacion entre una decisión y varias preguntas.
RESULTADO	Se añaden a la listaRelaciones todas las relaciones existentes entre la decisión y las preguntas.
ALGORITMO	1. **Obtener las clases** que pertenecen al rango de la propiedad *HayRespuestaNegativaEn* de la decisión (parámetro *clase*). 2. **Obtener las descripciones** de las clases del paso anterior, es decir, las **preguntas** en su forma abreviada. 3. Mientras haya preguntas por procesar: **Obtener identificador** (en la colección *listaPreguntas*) de la pregunta procesada. **Añadir el identificador a las preguntas relacionadas** con la decisión mediante métodos del objeto Relacion. Fin Mientras

Cada una de las clases que constituyen las preguntas tiene una instancia de sí misma, es decir, un individuo. Si observamos la Figura 22, la cual muestra la base de datos que manejan Pegaso y Gades, vemos como se almacenan dichos individuos. También se observa como están relacionados con una entrada correspondiente bien a un hito alarma o bien a un hito aviso, y éstos contienen la descripción completa de las preguntas.

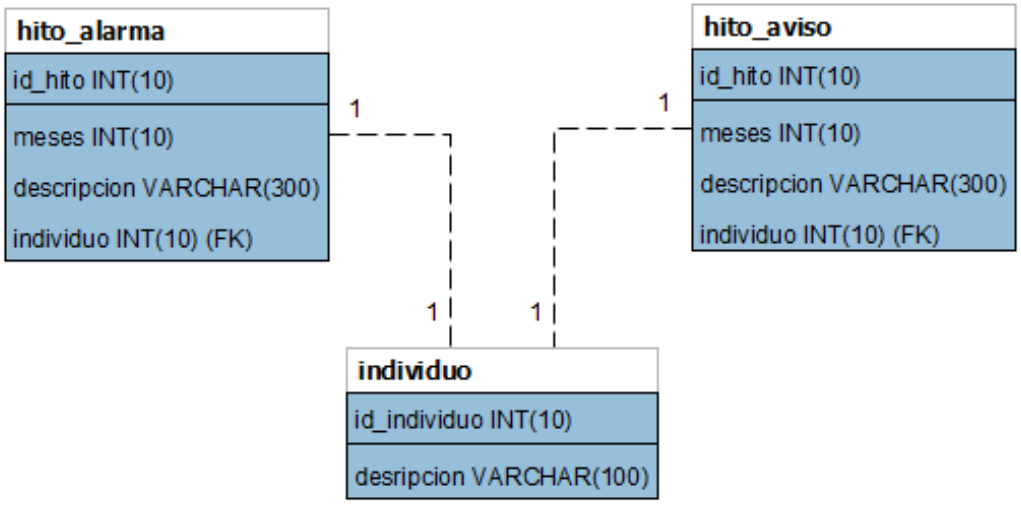


Figura 22: Extracto de la base de datos de Pegaso y Gades

Por lo tanto, para obtener la descripción completa de cada una de las preguntas será necesario utilizar el individuo obtenido del fichero OWL para encontrar la entrada de la base de datos de Pegaso y Gades que contiene dicha descripción completa.

Una vez obtenidos todos estos datos, podemos construir la base de datos que represente la información de la base de conocimiento (de Gades y Pegaso), es decir, las preguntas que constituyen los hitos, las decisiones del sistema y sus relaciones.

La Figura 23 nos muestra una visión de la base de datos de Galatea.

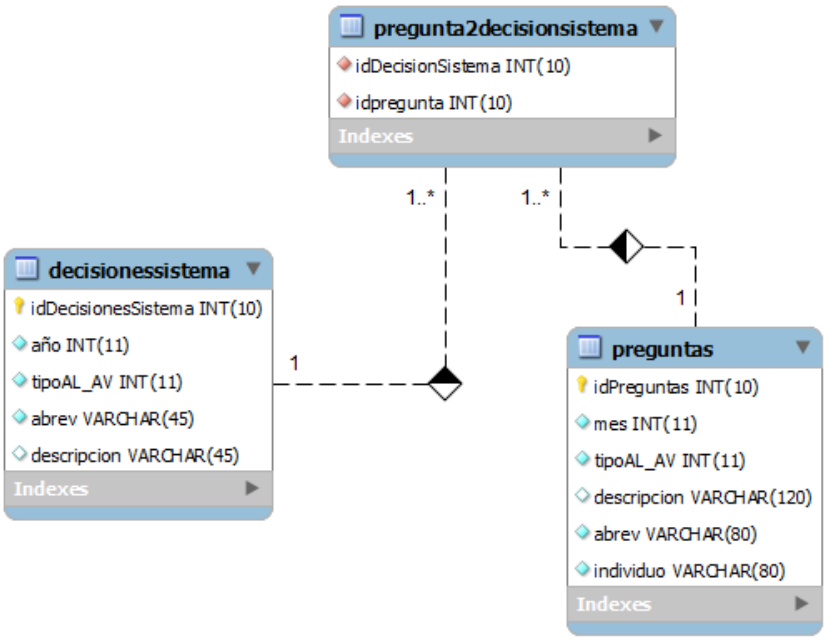


Figura 23: Modelo relacional de la base de datos de Galatea

De este modo, conseguimos almacenar incluso más información sobre una pregunta, pero en una sola tabla. A continuación se muestran dos ejemplos de los registros de las bases de datos. El primero se corresponde a Pegaso y Gades, donde la información mostrada está en las tablas *hito_alarma* e *individuo,* las cuales se han cruzado para juntar su contenido e ilustrar este ejemplo:

id_hito	meses	descripcion	id_individuo	descripcion
1	1	¿Reacciona a una campana?	1	#AL_I_NoReaccionaAUnaCampana
2	1	¿Vocaliza sin llorar?	2	#AL_I_NoVocalizaSinLlorar

Figura 24: Ejemplo de pregunta registrada en Pegaso o Gades

El segundo registro se corresponde a la base de datos de **Galatea,** y solo a su tabla *Preguntas:*

idPreguntas	mes	tipoAL_AV	descripcion	abrev	individuo
114	1	1	¿Reacciona a una campana?	AL_NoReaccionaAUnaCampana	AL_I_NoReaccionaAUnaCampana
115	1	1	¿Vocaliza sin llorar?	AL_NoVocalizaSinLlorar	AL_I_NoVocalizaSinLlorar

Figura 25: Ejemplo de pregunta registrada en Galatea

A continuación se van a describir más detalladamente las tablas y atributos que componen la base de datos para una mejor comprensión de la misma.

Tabla Preguntas

Esta tabla almacena las **preguntas** que realizan los sistemas Pegaso y Gades durante las evaluaciones. Estas preguntas están formadas por los siguientes atributos:

- **Identificador único** de la pregunta (clave primaria).
- **Mes** de la pregunta. Mes en el cual el hito que supone la pregunta debe haber sido superado por el niño evaluado.
- **Tipo** de hito. Establece si la no realización del hito de la pregunta supone una alarma o un aviso para el sistema.
- **Descripción** de la pregunta. Descripción completa, es decir, la pregunta tal cual se muestra en las evaluaciones.
- **Abreviatura o versión abreviada** de la pregunta. Como hemos visto anteriormente es la descripción o nombre de la clase correspondiente a la pregunta en el fichero OWL.
- **Individuo**. Al igual que la versión abreviada es un elemento necesario para el fichero OWL.

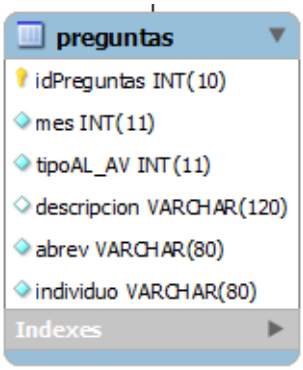


Figura 26: Tabla de preguntas

Los elementos "abreviatura" o "abrev" (en la tabla) e "individuo" de las preguntas serán necesarios cuando, tras la realización de cambios por parte de los expertos, se pretenda crear la ontología, el fichero OWL, automáticamente y se deban utilizar debido a las particularidades de estos atributos para formar parte del lenguaje aceptado por OWL, como no contener espacios o ser únicos en la ontología.

Tabla DecisionesSistema

A continuación, almacenamos las **decisiones del sistema**. Estas decisiones están formadas por:

- **Identificador único** de la decisión (clave primaria).
- **Año**. Al contrario que las preguntas, las decisiones están acotadas en función del año del niño al que pertenece la pregunta. Por ejemplo, una decisión perteneciente al año 3 será una decisión tomada en cualquier hito entre los meses 37 y 48.

- **Tipo**. Establece si la decisión supone una alarma o un aviso para el sistema.
- **Abreviatura** o versión abreviada de la decisión. Tiene unas características que la hacen compatible con OWL (eliminación de espacios, singularidad en el sistema). Al igual que en las preguntas, es utilizado en lo referente a la construcción del fichero OWL.

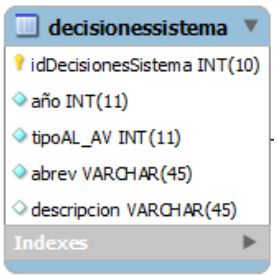


Figura 27: Tabla de decisiones del sistema

Por último, debemos establecer **las relaciones entre las preguntas y las decisiones**, lo que conseguimos mediante la siguiente tabla de referencias cruzadas. Esta tabla contiene para cada decisión del sistema, cuáles son las preguntas que tiene asociadas.

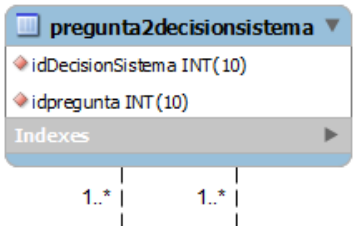


Figura 28: Tabla con las referencias cruzadas entre preguntas y decisiones del sistema

3.3.2. Almacenamiento de propuestas

Todo el sistema gira alrededor de las propuestas, por lo tanto es crucial el almacenamiento de todas estas propuestas de una manera correcta y fiable.

Esta tabla deberá guardar todos los datos que conforman una propuesta, es decir:

- **Identificador único** de la propuesta (clave primaria).
- **Descripción completa** de la pregunta. Pregunta completa tal cual aparecerá plateada durante las evaluaciones.
- **Mes** al que se corresponde la pregunta. Como anteriormente, se refiere al mes en el cual el hito debe haber sido superado.
- **Decisión del sistema** asociada a la pregunta.

- **Estado** de la propuesta. Indica la situación de la propuesta, si está siendo evaluada, si ha sido aceptada, etc.
- **Motivo** de la propuesta. Es el motivo por el cual se consideró erróneo el hito y se realiza la propuesta o si la propuesta es de nueva pregunta. Los posibles valores de este campo están predefinidos: "Nueva propuesta", si como dice la propuesta es completamente nueva; o expresando el motivo del cambio "Pregunta mal formulada/ambigua", "Imposible de contestar", "Edad no adecuada", "Decisión del sistema errónea/alarmante", "Otros".
- **Comentarios** que justifican la propuesta. Justificación del cambio propuesto, información añadida que pueden utilizar los expertos para evaluar la propuesta.
- **Usuario** que realizó la propuesta. Se almacena para filtrar la búsqueda durante la consulta del histórico de propuestas.

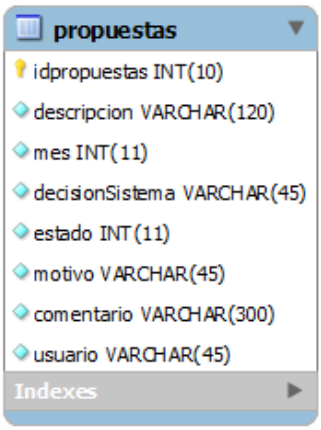


Figura 29: Tabla de propuestas

Esta tabla tiene una serie de particularidades, como se ha dicho anteriormente, una pregunta puede tener varias decisiones del sistema asociadas, pero la tabla solo almacena las decisiones de **una en una**. Por lo tanto, para una pregunta con múltiples decisiones se almacenarán varios registros con los mismos datos (descripción, mes, estado, etc.) y cada una de las decisiones seleccionadas. Esto permite que los expertos puedan evaluar las decisiones individualmente y no sólo a nivel de propuesta, ya que pueden considerar la propuesta parcialmente adecuada y aceptar solo un conjunto de las decisiones propuestas.

Del mismo modo, el estado de la propuesta no está siendo completamente explotado en la actualidad, ya que su fin es la modificación del mismo por parte de los expertos pasando del estado "Propuesta" inicial a otro estado tras su evaluación que informe al usuario sobre la situación de sus propuestas.

Como se explicaba en la Figura 16, este sistema colabora con otros sistemas y por ello existen atributos en esta base de datos que este sistema no va a utilizar con provecho, pero son necesarios para el conjunto de plataformas. En concreto, en la etapa de revisión/aceptación de propuestas por parte de los expertos, el sistema Galatea utilizará la información de ciertos campos de la base de datos que ahora no se están utilizando.

3.4. Diseño y construcción del acceso a datos

La capa de acceso a datos de la aplicación está formada por un conjunto de componentes que implementan los accesos a la base de datos y cuya misión es aislar dichos accesos de las capas de lógica y presentación.

En este apartado se van a describir tanto los objetos de acceso a datos (DAO), que son los componentes que forman esta capa, como los objetos de transferencia de datos, ya que su origen se sitúa en esta capa aunque sean utilizados por la capa de lógica.

3.4.1. Objetos de Acceso a Datos

Los componentes Java que se utilizan en esta capa para acceder a los datos siguen el patrón de Objeto de Acceso a Datos (DAO).

Según la definición de este patrón: "*El DAO separa una interfaz cliente de recursos de datos de sus mecanismos de acceso y adapta un API de acceso a datos a una interfaz cliente genérica*" [ALU01].

Esto significa que el DAO debe aislar los métodos de lógica, que necesitan los datos, del API utilizado para el acceso a los mismos.

En el caso de nuestro sistema el API seleccionado para el acceso a datos es Java Data Base Connectivity (JDBC). Este API permite la ejecución de sentencias SQL sobre una base de datos a partir de una serie de objetos. Estos objetos son:

- El objeto que representa la **conexión** con la base de datos.
- El objeto que representa la **sentencia** a ejecutar sobre la base de datos.
- El objeto que representa el **conjunto** de resultados generados con la ejecución de la sentencia en el caso de que la sentencia sea una consulta.

En la aplicación desarrollada se han construido dos objetos de acceso a datos. Estos objetos están implementados en las clases:

- *galatea.data.**EvolutionDAO***: encapsula todos los métodos de acceso a la base de datos necesarios para los procesos de realización de propuestas y consulta de las mismas
- *galatea.data.**AutenticarDAO***: encapsula los métodos de acceso a la base de datos de los sistemas Pegaso y Gades para la autenticación de los usuarios.

Se ha desarrollado además una clase llamada *galatea.data.**BasicDAO*** cuya función será contener todos los métodos comunes al resto de DAOs. Mediante el mecanismo de reutilización de código existente en los lenguajes orientados a objetos denominado *herencia* se consigue que las clases EvolutionDAO y AutenticarDAO hereden el código definido en BasicDAO.

El código heredado de BasicDAO es el relacionado con la obtención y liberación de las conexiones con la base de datos. Por lo tanto, las clases EvolutionDAO y AutenticarDAO implementarán los métodos de acceso a datos, delegando en los métodos de BasicDAO la obtención y liberación de conexiones. La Figura 30 ilustra la relación comentada:

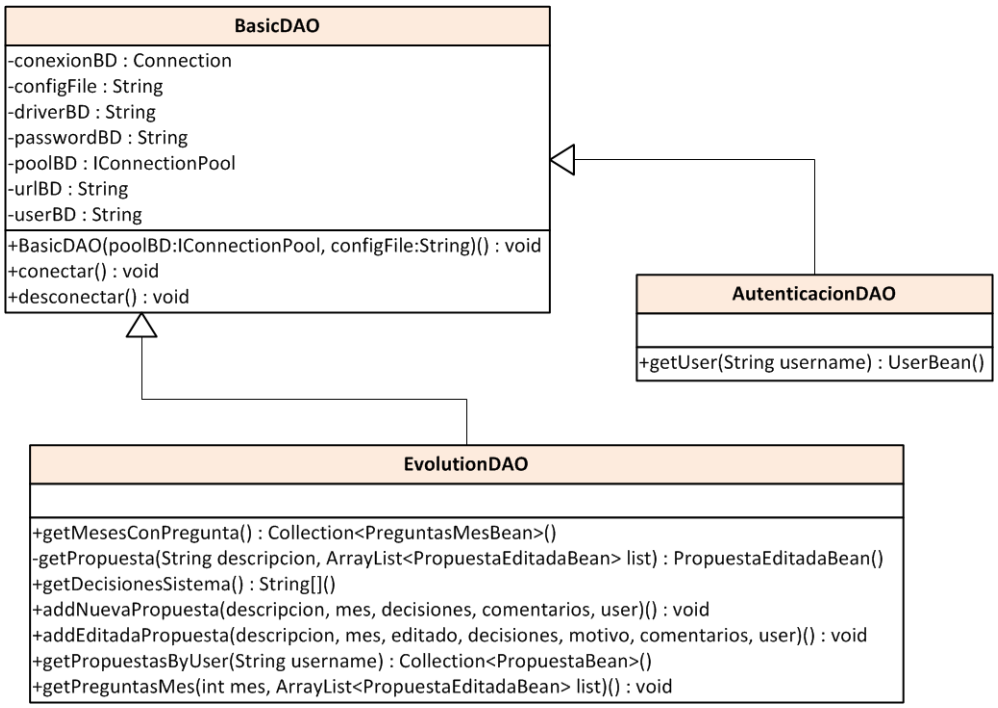


Figura 30: Jerarquía de clases DAO

Los tipos de datos no se detallan en algunos métodos por simplificar la ilustración y facilitar su comprensión, ya que dichos tipos no son necesarios para el correcto entendimiento de la función que ejercen.

Gestión de las conexiones con la base de datos

Tal y como se acaba de comentar, la aplicación desarrollada utiliza una **reserva o pool de conexiones**.

Un pool de conexiones se implementa mediante un objeto que incluye algún tipo de almacén a partir del cual se van obteniendo las conexiones a medida que un cliente las solicita. A dicho almacén se devuelve la conexión una vez utilizada para que otros clientes puedan usarla. Si el almacén no tiene conexiones en el momento en el que un cliente la solicita, el objeto pool se encarga de crear una nueva conexión y devolverla al cliente.

En el caso de nuestra implementación del pool, el objeto utilizado para almacenar las conexiones es una pila (java.util.Stack). La clase donde se ha implementado el pool es *galatea.data.**ConexionPoolBasico***. Esta clase dispone de métodos para que los DAO obtengan y devuelvan conexiones a la base de datos. Los DAO actúan como clientes del pool.

Los objetos de acceso a datos disponen de los métodos heredados de BasicDAO que permiten obtener y devolver una conexión del pool:

- Método conectar(): obtiene una conexión del pool.
- Método desconectar(): devuelve una conexión al pool.

Métodos de manejo de datos

Los DAO dispondrán de métodos de manejo de los datos de la base de datos. En este sistema existirán dos tipos de métodos:

- Los **métodos de consulta** encapsulan la ejecución de sentencias de tipo SELECT de SQL, es decir, búsquedas. Estas sentencias pueden estar parametrizadas, es decir, podrán recibir valores que modifiquen el criterio de búsqueda utilizado.
  En estos objetos se realizará el tratamiento de los resultados de la sentencia devolviendo dichos resultados como un objeto de transferencia de datos o como una colección de los mismos dependiendo del tipo de consulta. Si se realiza mediante una clave única de una tabla, la respuesta será un solo objeto identificado por esa clave; si la sentencia ejecuta criterios de búsqueda que no incluyen clave única, la respuesta será una colección de objetos que cumplen dichos criterios.
- Los **métodos de inserción** de datos encapsulan la ejecución de sentencias INSERT de SQL, las cuales como la propia palabra indica “insertan” los registros en la base

de datos. Estos métodos deben recibir como parámetros los datos que deben incorporar a la base de datos.

3.4.2. Objetos de transferencia de datos

Los **objetos de transferencia** de datos son esenciales en el tratamiento del acceso a datos del sistema, ya que permiten el intercambio de información entre capas mediante objetos que simplifican estas tareas.

Estos objetos son instancias de clases Java que siguen ciertas reglas de definición que permiten su reutilización y facilitan el acceso desde los diferentes componentes de la aplicación. Estos objetos son los denominados **Java Beans**. Cada instancia representa bien un registro de una tabla de la base de datos, o bien una unidad de información con unos atributos fijos y que es utilizada por el sistema.

Estos Java Beans contendrán una serie de atributos, que podrán ser modificados mediante métodos denominados "setters" y consultados mediante métodos denominados "getters".

```
public Integer getIndividuo(){
    return this.individuo;
}

public void setIndividuo(Integer individuo){
    this.individuo=individuo;
}
```

Figura 31: Ejemplo de "getters" y "setters" de un Java Bean

Debido a las necesidades del sistema se han establecido los siguientes objetos Java Bean que atienden a las diferentes necesidades para las que han sido creados:

- *galatea.data.**PreguntasMesBean**:* contiene un mes y el número de preguntas existentes para dicho mes. Es utilizado por el sistema en el trascurso de la búsqueda de las preguntas que se desea editar. Su uso recurrente justifica la creación de este objeto específico.
- *galatea.data.**PropuestaNuevaBean***: contiene los datos acerca de una propuesta nueva de pregunta.
- *galatea.data.**PropuestaEditadaBean***: contiene los atributos que representan una propuesta de cambio sobre una pregunta existente.

- *galatea.data.**PropuestaBean***: contiene los atributos sobre una propuesta contenida en la base de datos, por lo tanto, existe un mapeo directo con la tabla *Propuestas* de la base de datos.
- *galatea.data.**UserBean***: representa un usuario de Pegaso o Gades por su usuario y contraseña. Se utiliza para la autenticación de los usuarios.

3.5. Diseño y construcción de la lógica funcional o de negocio

La capa de lógica funcional o de negocio de la aplicación tiene dos propósitos fundamentales:

- ❖ Implementar los algoritmos lógicos asociados a las funcionalidades: lógica del sistema.
- ❖ Gestionar el intercambio de información entre la interfaz de usuario y la base de datos: intercambio de datos.

Esta capa en la aplicación está formada por componentes Java Beans que encapsulan los métodos de lógica del sistema y los métodos de intercambio de datos.

Los componentes de esta capa son utilizados por los ayudantes de vista de la capa de presentación (este patrón y su implementación en el sistema se explican con más detalle más adelante en el apartado "Diseño y construcción de la interfaz de usuario"), por otro lado, invocan a los métodos de los objetos de acceso a datos de la capa descrita en el apartado anterior.

El diseño de estos componentes se ha realizado aplicando dos patrones de diseño:

- **Patrón delegado de negocio**: que "*desacopla los componentes de negocio del código que los usa, gestionando la complejidad de la búsqueda de componentes distribuidos y el manejo de excepciones y puede adaptar la interfaz de los componentes de negocio a una interfaz más simple utilizada por las vistas*" [ALU01].
- **Patrón manejador de lista de valores**: que "*proporciona una forma más eficiente de iterar sobre grandes listas de sólo lectura a través de las diferentes capas*" [ALU01].

En los siguientes apartados se va a proceder a describir el diseño de estos componentes de la capa de negocio detallando las características más interesantes de los mismos desde el punto de vista de la implementación.

3.5.1. Componentes delegados de negocio o controladores de lógica

Los componentes que siguen el patrón de *delegado de negocio*, también nos referiremos a ellos como *controladores de lógica*.

La utilización de este patrón se contempla a la hora de, como se indica en la definición del apartado anterior, "gestionar la complejidad" de la propia lógica. En este caso, la complejidad del sistema desde el punto de vista lógico no era muy grande, y por lo tanto la utilización del patrón no resulta imprescindible, pero sí recomendable para mejorar la adaptación del sistema a posibles cambios.

Por ello, se han utilizado tres componentes siguiendo este patrón:

- *galatea.logic.**ConsultaDatosController***: se encarga de las consultas sobre la base de datos con los criterios y en las formas requeridas.
- *galatea.logic.**AlmacenarPropuestasController***: se encarga de la transmisión de la información sobre las propuestas hacia la base de datos.
- *galatea.logic.**AutenticacionController***: se encarga de la autenticación de los usuarios en el sistema.

En todos ellos encontramos métodos de intercambio de datos, donde la lógica que contienen es mínima y, por lo tanto, no reseñable.

La mayoría de los métodos de los componentes de la capa de lógica actúan como intermediarios entre los ayudantes de vista y un objeto de acceso a datos.

Los métodos del controlador de lógica realizan las siguientes funciones:

- **Envuelven las llamadas** a los métodos del DAO devolviendo objetos de transferencia de datos, descritos en el apartado 3.4.2.
- **Atrapan las excepciones** del DAO y las convierten a otros tipos de excepciones comprensibles por la capa de presentación.
- **Registran toda esta actividad** en un archivo de bitácora (log) disponible en el contenedor web Tomcat.

A continuación se muestran los diferentes métodos que contienen cada uno de los controladores de lógica comentados.

En primer lugar, *galatea.logic.**ConsultaDatosController,*** el cual facilita a la capa de presentación el acceso a los datos que más tarde se mostrarán al usuario. Los métodos se muestran en la Tabla 2.

Tabla 2: Métodos del controlador de lógica ConsultaDatosController

Tipo de retorno	Método y descripción
String[]	**getDecisionesSistema** ( ) Este método se utiliza para obtener las decisiones del sistema que son posibles, es decir, que pueden ser resultado de una evaluación en Pegaso o Gades.
Collection <PreguntasMesBean>	**getNumeroPreguntasMeses** (int anyo) Este método se utiliza para obtener los meses de un año con preguntas y cuantas preguntas tiene.
Collection <PropuestaEditadaBean>	**getPreguntasxMeses** (int[] iMeses) Este método se utiliza para obtener las preguntas correspondientes a un determinado mes
Collection <PropuestaBean>	**getPropuestasByUser** (java.lang.String usuario) Este método se utiliza para obtener todas las propuestas realizadas por un usuario determinado

En segundo lugar, *galatea.logic.**AlmacenarPropuestasController***, el cual encapsula los métodos que utilizarán los ayudantes de vista para almacenar las propuestas capturadas mediante las vistas. Estos métodos se muestran en la Tabla 3.

Tabla 3: Métodos del controlador de lógica AlmacenarPropuestasController

Tipo de retorno	Método y descripción
void	**createNuevaPregunta** (PropuestaNuevaBean nueva, String user) Este método se encarga de incluir una propuesta de nueva pregunta en la base de datos.
void	**createPropuestaEditada** (PropuestaEditadaBean prop, String user) Este método se encarga de incluir una propuesta de cambio en la base de datos.

Por último, *galatea.logic.**AutenticacionController***, el menos complejo de los controladores, está encargado de encapsular el acceso a las bases de datos de Pegaso y Gades para obtener los datos de un usuario cuando intenta autenticarse en el sistema. Esta operación se realiza mediante el método descrito en la Tabla 4.

Tabla 4: Método del controlador de lógica AutenticacionController

Tipo de retorno	Método y descripción
data.UserBean	**getUserByName**(java.lang.String username) Este método se encarga de obtener el usuario y la contraseña de la base de datos de Pegaso o Gades para su autenticación.

3.5.2. Manejador de lista de propuestas

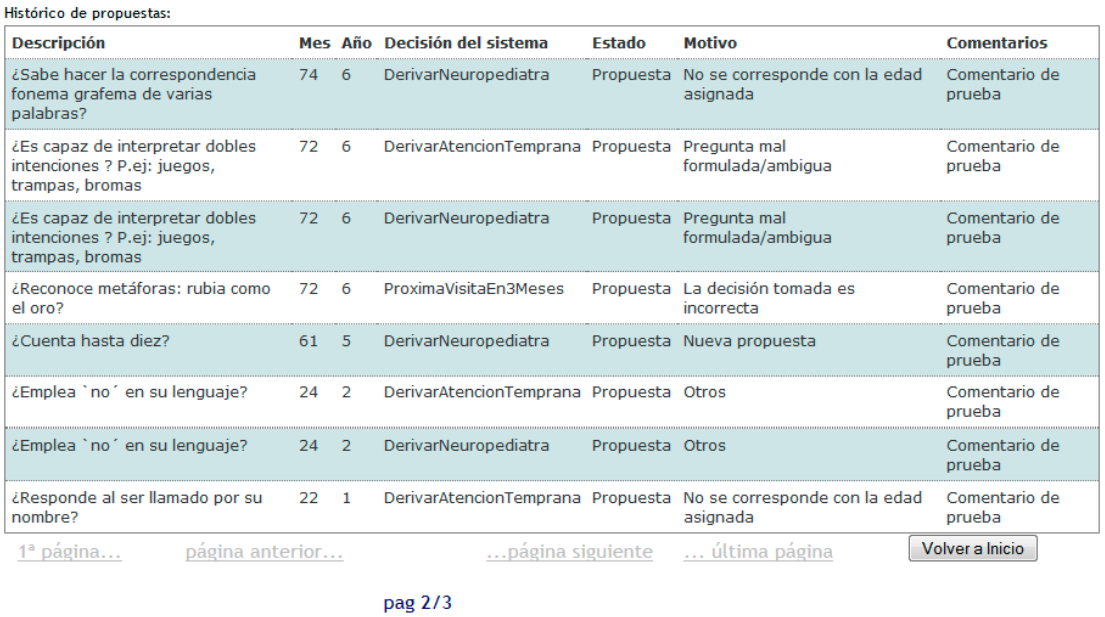
Una vez descritos los controladores de lógica usados en la aplicación se va a pasar a describir el otro componente de lógica utilizado en la aplicación: el manejador de la lista de propuestas. Este manejador está implementado en la clase ListPager. Su propósito fundamental es gestionar la paginación e iteraciones sobre la colección de datos que devuelve el controlador *ConsultaDatosController* cuando se le solicita el histórico de propuestas de un usuario.

Para ello, esta clase define una serie de atributos privados que le permiten controlar la paginación, como son el número de páginas totales, el número de datos por página y el índice de la página actual.

Añade a estos atributos una serie de métodos que permiten:

- Obtener las diferentes páginas de datos y moverse por ellas.
- Saber si la página actual es la primera o la última.
- Actualizar los datos de la lista a partir de una nueva consulta.
- Consultar y establecer los valores de los atributos privados de control.

Este componente ayudará a la vista y al ayudante de la vista encargados de la consulta del histórico de propuestas de un usuario, facilitando la tarea del manejo y paginación en el caso de que existan muchas propuestas (tras el uso durante un período considerable de la plataforma).



Histórico de propuestas:

Descripción	Mes	Año	Decisión del sistema	Estado	Motivo	Comentarios
¿Sabe hacer la correspondencia fonema grafema de varias palabras?	74	6	DerivarNeuropediatra	Propuesta	No se corresponde con la edad asignada	Comentario de prueba
¿Es capaz de interpretar dobles intenciones ? P.ej: juegos, trampas, bromas	72	6	DerivarAtencionTemprana	Propuesta	Pregunta mal formulada/ambigua	Comentario de prueba
¿Es capaz de interpretar dobles intenciones ? P.ej: juegos, trampas, bromas	72	6	DerivarNeuropediatra	Propuesta	Pregunta mal formulada/ambigua	Comentario de prueba
¿Reconoce metáforas: rubia como el oro?	72	6	ProximaVisitaEn3Meses	Propuesta	La decisión tomada es incorrecta	Comentario de prueba
¿Cuenta hasta diez?	61	5	DerivarNeuropediatra	Propuesta	Nueva propuesta	Comentario de prueba
¿Emplea `no´ en su lenguaje?	24	2	DerivarAtencionTemprana	Propuesta	Otros	Comentario de prueba
¿Emplea `no´ en su lenguaje?	24	2	DerivarNeuropediatra	Propuesta	Otros	Comentario de prueba
¿Responde al ser llamado por su nombre?	22	1	DerivarAtencionTemprana	Propuesta	No se corresponde con la edad asignada	Comentario de prueba

Figura 32: Vista de la tabla de consulta del histórico de propuestas

Esta vista obtiene los datos a mostrar del manejador ListPager y usa sus métodos para moverse por la colección de propuestas realizadas y generar la tabla que presenta al usuario. Puede observarse en la figura anterior los enlaces debajo de la tabla que permiten la iteración sobre los datos.

En la Tabla 5, extraída de la documentación Java del proyecto, que se muestra a continuación puede verse la relación de métodos que forman parte de este Java Bean:

Tabla 5: Métodos del manejador ListPager

Tipo de retorno	Método y descripción
void	**actualizar**(java.util.Collection datos) Actualiza los datos del paginador, el número de páginas e inicializa el puntero de páginas.
java.util.Collection	**getActualPage**() Obtiene la página de datos actual.
java.util.Collection	**getDatos**() Obtiene una colección con todos los datos que tiene el paginador asociados a la consulta actual.
java.lang.Short	**getDatosPorPagina**() Obtiene el número de datos por página que se mostrarán.
java.util.Collection	**getFirstPage**() Obtiene la primera página de datos.ATENCIÓN: mueve el puntero de página actual a la primera página.
java.util.Collection	**getLastPage**() Obtiene la última página de datos.
java.util.Collection	**getNextPage**() Obtiene la página de datos siguiente.

Tipo de retorno	Método y descripción
java.lang.Integer	**getNumPaginas**() Obtiene el número de páginas de datos para la consulta actual del paginador.
java.lang.Integer	**getPagActual**() Obtiene el número de la página actual de datos que proporciona el paginador.
java.util.Collection	**getPrevPage**() Obtiene la página de datos anterior.
java.lang.String	**getValorCriterio**() Obtiene el valor del criterio de consulta.
java.lang.Boolean	**isFirstPage**() Indica si la página actual es la primera página de datos.
java.lang.Boolean	**isLastPage**() Indica si la página actual es la última página de datos.
void	**setDatos**(java.util.Collection datos) Establece la colección de datos asociados a la consulta actual.
void	**setDatosPorPagina**(java.lang.Short datosPorPagina) Establece el número de datos por página que se mostrarán.
void	**setPagActual**(java.lang.Integer pagActual) Establece el número de la página actual de datos que proporciona el paginador.
void	**setValorCriterio**(java.lang.String valorCriterio) Establece el valor del criterio de consulta.

3.6. Diseño y construcción de la interfaz de usuario

La capa de interfaz de usuario o de presentación está constituida por dos subcapas, tal y como se ha comentado en apartados anteriores:

- La capa web del lado **cliente**, donde se interpretan y muestran las vistas de usuario.
- La capa web del lado **servidor**, donde se realiza el procesamiento correspondiente a la capa de presentación.

Los componentes que se utilizarán para llevar a cabo estas funciones se dividirán en:

- **Vistas**: implementadas mediante Java Server Pages (JSP), actúan como Interfaz Gráfico de Usuario (IGU).
- **Ayudantes de vistas**: implementados mediante Servlets, actúan como ayudantes de las vistas procesando las peticiones originadas por las vistas y preparando los datos de la respuesta de manera que estén accesibles para las vistas.

Las vistas y los ayudantes de vistas compartirán información mediante un objeto denominado **sesion**. Este objeto sesión será gestionado por el contenedor web y estará asociado a cada una de las sesiones abiertas en el sistema por los usuarios del mismo, permitiendo mantener un contexto compartido.

En los siguientes apartados van a describirse estos componentes de la capa de presentación y su interacción.

3.6.1. Ayudantes de vistas

Los ayudantes de vistas "*son clases que realizan la recepción de datos para la vista y gestionan el procesado adicional necesario para la misma*" [ALU01]. Con el uso de estos ayudantes de vista, se permite a los JSP centrarse en la presentación y su lógica.

Además el uso de este patrón permite desacoplar la capa de lógica de negocio de la de presentación, haciendo que los cambios que se pueden producir en la capa de lógica y los aún más frecuentes cambios que se puedan producir en las vistas no constituyan un problema desde el punto de vista del mantenimiento y la reutilización.

En la Figura 33 se muestra el diagrama de clases correspondiente a los componentes de la capa de presentación para el proceso de propuesta de cambios.

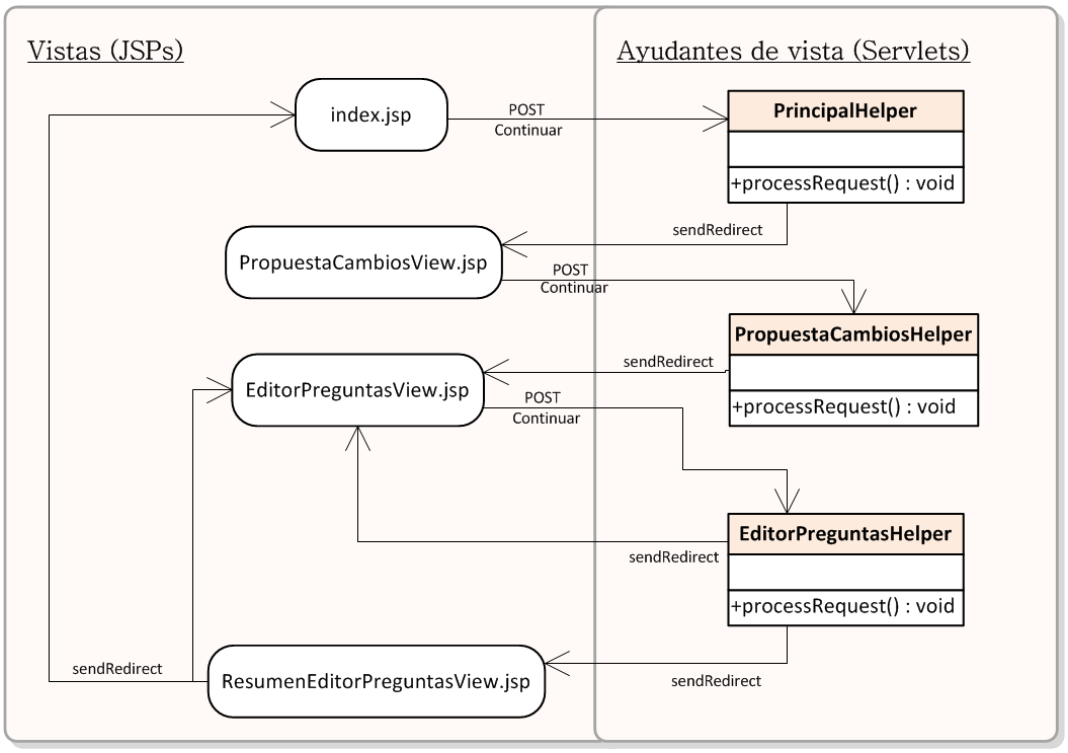


Figura 33: Componentes de la capa de presentación para el proceso de propuesta de cambios

Como se observa en la Figura 33, cada uno de los procesos que puede realizar el sistema (los casos de uso) está descompuesto en etapas, las cuales se realizan mediante parejas de vista-ayudante de vista. De este modo, conseguimos que las funcionalidades de mayor nivel sean implementadas mediante una secuencia encadenada de vistas y ayudantes de vista, cada uno de ellos con una responsabilidad dentro del proceso.

Una pareja vista-ayudante de vista particular es la formada por *EditorPreguntasView.jsp* y *EditorPreguntasHelper.java*, ya que entre ellos se realiza toda la función de edición de las preguntas seleccionadas anteriormente. Esta edición se realiza pregunta a pregunta, por lo que, tras enviar los cambios realizados en una pregunta y ser procesador por el ayudante de vista, se vuelve a la misma vista para continuar con la edición del resto de preguntas. Es decir, se produce un bucle entre ambos hasta que se finaliza la edición.

Esta secuencia se muestra en la Figura 34.

Los bucles comentados pueden observarse en la misma figura entre la vista *EditorPreguntasView* y el ayudante de vista *EditorPreguntasHelper*.

Así pues los ayudantes de vista:

- Recogen los datos enviados desde una vista y los validan.
- Procesan estos datos, realizando los accesos necesarios a la información de sesión y de la capa de negocio.
- Preparan los datos necesarios para la siguiente vista, dirigiendo la visualización del usuario a dicha vista.

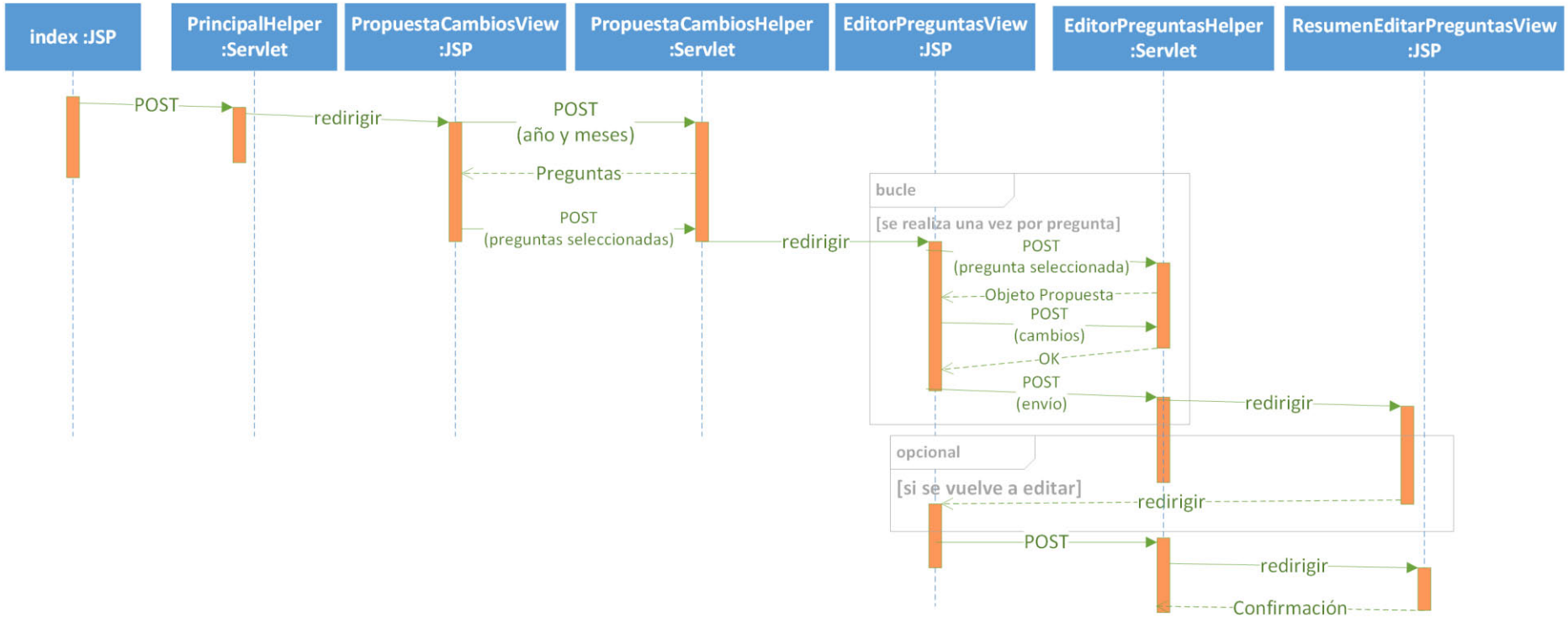


Figura 34: Interacción de componentes de la capa de presentación en el proceso de propuesta de cambios

3.6.2. Vistas de usuario

Las vistas son "*cualquier representación de datos que se utilizan para mostrar información a un usuario*" [WIK13]. Su utilidad no se reduce simplemente a la presentación de información sino que además permiten capturar las acciones y la información aportada por el usuario.

En nuestro sistema, los *ayudantes de vista* procesan las peticiones y añaden los datos necesarios a la *sesión*, de donde las vistas de usuario obtienen la información y la presentan con la organización y aspecto deseado. En la Figura 35 se muestra el diagrama de clases para los componentes de la capa de presentación de la realización de propuestas de nuevas preguntas.

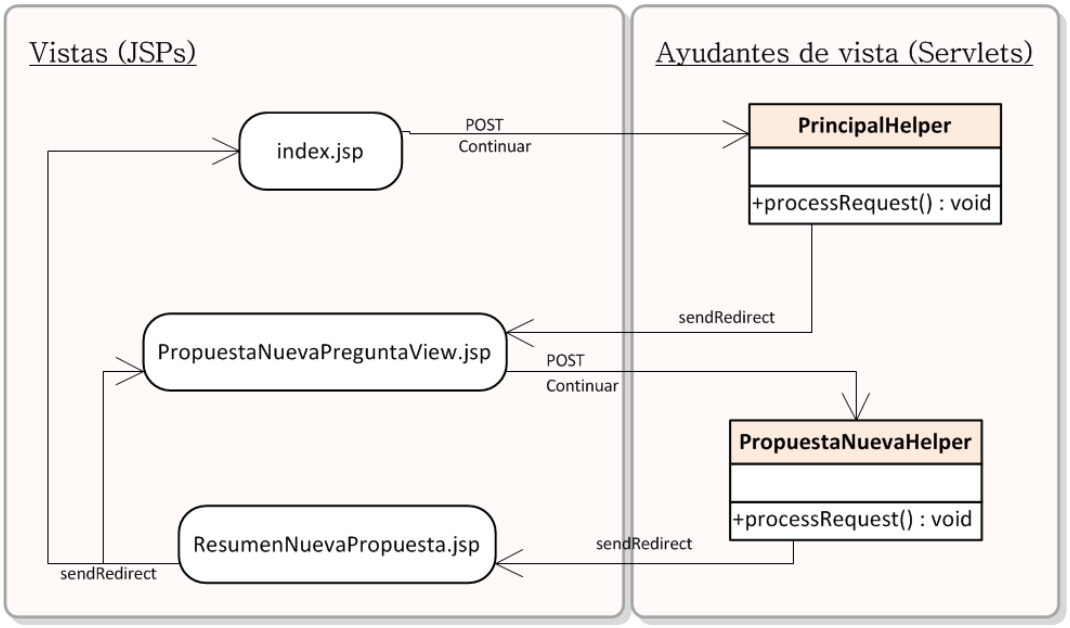


Figura 35: Componentes de la capa de presentación para el proceso de nuevas propuestas

En las figuras 33 y 35 podemos observar como las vistas son páginas web dinámicas Java de servidor: JSP. El acceso de las vistas a la información que necesitan presentar, datos almacenados por los ayudantes de vista en la sesión, se realiza utilizando los elementos constructivos propios de JSP.

Las vistas realizan pequeños procesados lógicos en el lado del servidor fundamentalmente orientados a la forma de presentar los datos. Las vistas JSP se traducen en el lado servidor generando una respuesta enviada al cliente web en forma de XHTML y Javascript (como la interacción mostrada con las figuras 36 y 37).

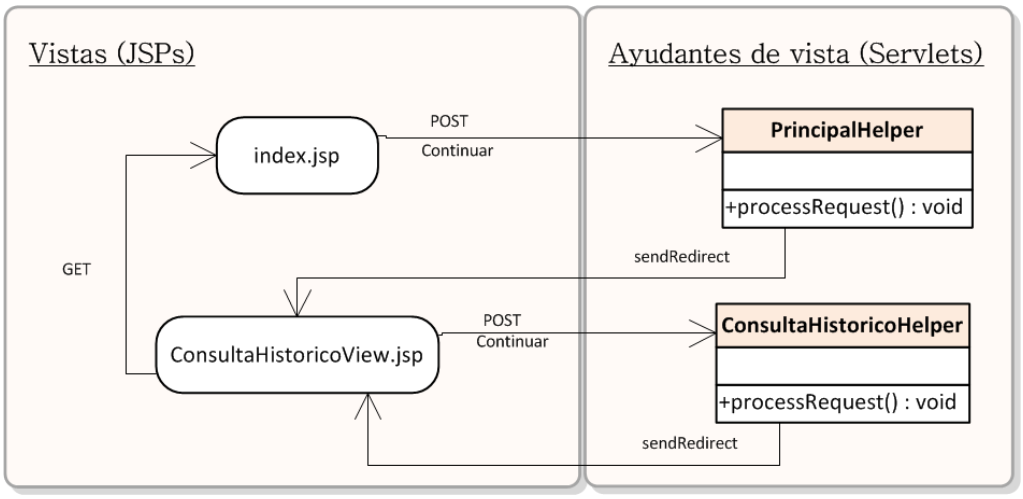


Figura 36: Componentes de la capa de presentación para la consulta de propuestas

El XHTML se interpreta en el navegador web y con la ayuda de las hojas de estilo en cascada (CSS) se presenta al usuario la vista con el aspecto definido.

El código Javascript se procesa en el lado cliente y se utiliza para implementar comportamientos dinámicos del lado del cliente y realizar determinadas validaciones de datos en el lado cliente.

La mayor parte de las páginas están orientadas a guiar al usuario en la realización de propuestas, por lo que la recolección de datos mediante formularios es constante, alternándolos con la presentación de información adicional.

Realizar un diseño del proceso, dividido en pequeñas actividades realizadas por los usuarios mediante las vistas, hace que las páginas visualizadas no sean muy complejas y por lo tanto de fácil mantenimiento y evolución.

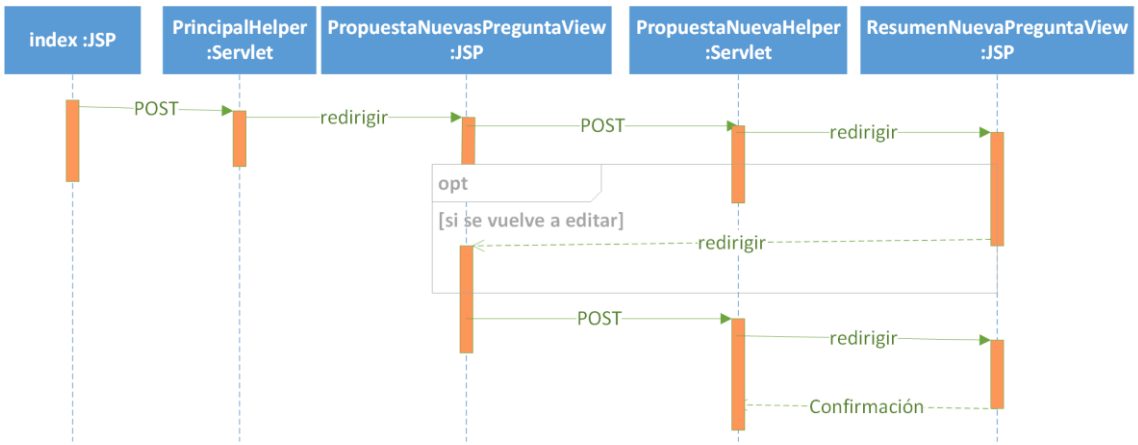


Figura 37: Interacción de componentes en el proceso de propuesta de nueva pregunta

Elementos constructivos adicionales en las vistas: etiquetas personalizadas

En cuanto a los detalles de implementación de las vistas, como elementos constructivos adicionales se han desarrollado etiquetas *JSP personalizadas de utilidad*.

Las etiquetas personalizadas son etiquetas con sintaxis XML que se declaran en un archivo descriptor de etiquetas y que tienen como soporte una clase Java, a la que se suele referir como manejador de etiqueta (*TagHandler*), que contiene código Java que se ejecuta cuando la etiqueta se traduce dentro de una página JSP. Concretamente se ha creado una etiqueta:

<evo:encodeUrl>: Se encarga de codificar una URL con el identificador de sesión de forma que si un cliente deshabilita las cookies se pueda seguir realizando el seguimiento de sesión. El seguimiento de sesión permite al contenedor web devolver a los componentes web el objeto sesión asociado a una interacción con un usuario determinado. Se utiliza en todas las vistas donde hay una URL destino de hiperenlace o de acción de formulario.

3.6.3. El contexto de colaboración entre ayudantes y vistas: la sesión

Como se ha comentado los ayudantes de vista colaboran con las vistas de usuario proporcionando a estas toda la información que necesitan mediante un objeto adicional: la sesión.

El objeto sesión tiene la función de almacenar toda la información que necesiten los componentes web, relacionada con una sesión interactiva de un usuario. Es decir, cada usuario que interactúa con la aplicación tiene asociada, en el lado del servidor, una información relacionada con su actividad. Esta información se almacena y se recupera del objeto sesión por parte de los componentes web de la capa de presentación.

El objeto sesión almacena la información en forma de múltiples objetos, entre los que se encuentran Java Bean y colecciones de Java Bean. Cada uno de estos objetos, almacenados en la sesión, tiene asociado un nombre único que permitirá el acceso al mismo. A la pareja nombre de objeto-objeto que se almacena en la sesión se le denomina atributo de sesión.

En la Tabla 6 se muestran todos los atributos de sesión que maneja la aplicación desarrollada:

Tabla 6: Atributos de sesión

Nombre de atributo	Tipo y descripción
decisiones	**String[]** Array que contiene las posibles decisiones del sistema.
userprop	**Collection<PropuestaBean>** Colección con el histórico de propuestas del usuario autenticado.
user	**String** Nombre del usuario autenticado.
estado	**String** Estado de la sesión.
colPropEdit	**Collection<PropuestaEditadaBean>** Colección de propuestas seleccionadas para editar o en proceso de edición.
numPreg	**int** Número de la pregunta en edición dentro de la colección de propuestas.
propEdit	**PropuestaEditadaBean** Propuesta en edición actualmente cuando se están editando un grupo de preguntas.
colPropSesion	**Collection<PropuestaBean>** Colección de propuestas realizadas por el usuario en la sesión actual.
propuesta	**PropuestaNuevaBean** La propuesta de nueva pregunta realizada.
anyo	**int** Año seleccionado para mostrar sus preguntas.
cPregMeses	**Collection<PreguntasMesBean>** Colección de objetos PreguntasMesBean para el año seleccionado en la búsqueda.
colPreg	**Collection<PropuestaEditadaBean>** Colección que contiene las preguntas de los meses seleccionados en la búsqueda.

3.6.4. Acceso a la aplicación y cierre de sesión

La aplicación web desarrollada tiene una vista de bienvenida o de índice (*index.jsp*) la cual solicita la autenticación del usuario para poder realizar cualquier otra operación. Esta autenticación se apoya en un *ayudante de vista* especializado (*AutenticacionHelper*) el cual redirigirá de nuevo a la vista anterior, ahora sí permitiendo realizar operaciones si la autenticación es positiva o mostrando un error de no ser así. Esta interacción se muestra en la siguiente figura:

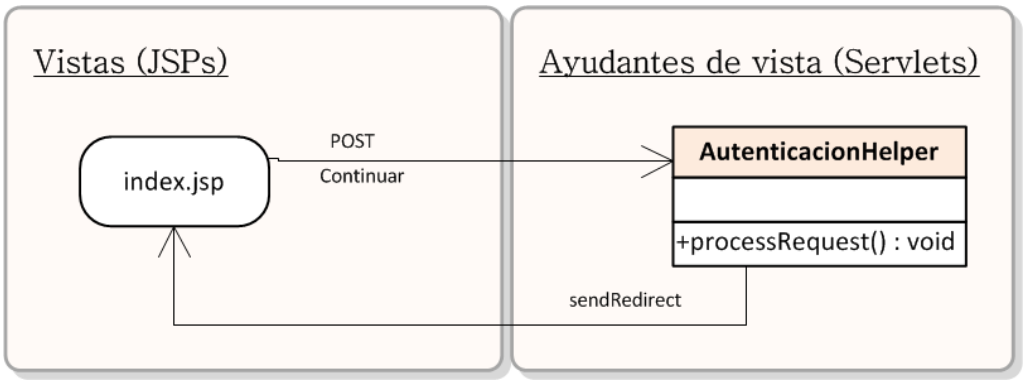


Figura 38: Componentes del proceso de autenticación

Por último en la Figura 39 se muestra la estructura de la operativa de cierre de sesión de los usuarios. En esta operativa el ayudante de vista *LogOutHelper* simplemente se encargará de invalidar la sesión y dirigir la visualización del usuario a la página de índice de la aplicación (*index.jsp*) cuando reciba la petición de salir de cualquier vista.

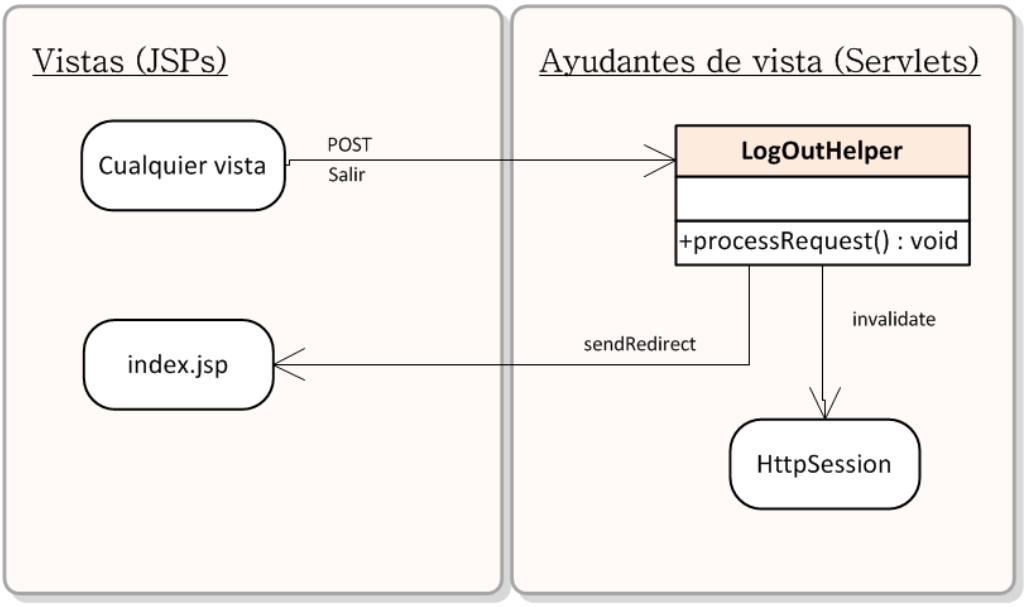


Figura 39: Componentes del proceso de cierre de sesión

3.7. Gestión de errores

La gestión de errores de la aplicación se basa en el tratamiento de las excepciones Java. Las excepciones son objetos Java, de una determinada clase, que son lanzados por las aplicaciones cuando se producen condiciones de error o bien cuando se producen situaciones excepcionales en el procesado normal de una aplicación.

Estas excepciones pueden ser atrapadas por los diferentes componentes de la aplicación o, en última instancia, por el contenedor web.

En la implementación del sistema se han definido diferentes excepciones asociadas a situaciones que se pueden producir en las diferentes capas de procesamiento.

Tabla 7: Excepciones del sistema

Excepción	Descripción
AutenticacionException	Error de autenticación de usuario.
AlmacenarPropuestaException	Error durante el almacenamiento de propuestas.
ConsultaDatosException	Error durante la consulta de datos.
PropuestaInvalidaException	Error en los atributos que conforman una propuesta.
DAOException	Error en el acceso a datos, normalmente SQLException.
SessionException	Error que indica que no existe sesión valida.

Cuando se producen excepciones se realizan dos tipos de tratamientos en la aplicación:

- **Se atrapan las excepciones y se tratan**: En este caso los componentes de lógica de la aplicación y los ayudantes de vista atrapan las excepciones, dejan un registro en los archivos de bitácora (log) de la aplicación y realizan una serie de acciones posteriormente. Otras variantes de este tipo de tratamiento convierten el tipo de la excepción y propagan la nueva excepción, consiguiendo encauzar las excepciones para tratarlas como se crea oportuno.
- **Se propagan las excepciones hasta el contenedor web**: En el caso de no atrapar una excepción esta se propaga hasta el contenedor web.
  El contenedor web entonces comprobará si nuestra aplicación web define las páginas de error que se deben enviar al cliente web asociadas a cada excepción. Si hay una página de error asociada a la excepción, se muestra dicha página. Si no existe esta asociación se muestra la página de error por defecto del contenedor web.

Se puede definir una página de error asociada a códigos de error del protocolo HTTP además de a las excepciones de nuestra aplicación. La Tabla 8 muestra las asociaciones definidas para las páginas de error de nuestra aplicación.

Tabla 8: Páginas de error y excepciones de la aplicación

Página de error	Código de error	Tipo de excepción
/ErrorView.jsp		galatea.logic.ConsultaDatosException
/ErrorView.jsp		galatea.logic.AlmacenarPropuestaException
/ErrorView.jsp		galatea.interfaz.PropuestaInvalidaException
/index.jsp		galatea.interfaz.AutenticacionException
/ErrorView.jsp		galatea.interfaz.SessionException
/ErrorView.jsp	500	Errores internos del servidor

A modo de ejemplo, puede observarse (en la siguiente figura) la página de error *ErrorView.jsp* asociada a la excepción *SessionException*.

Figura 40: Página de error para SessionException

3.8. Archivos de configuración

Los archivos de configuración de la aplicación tienen el propósito de permitir cambios en ciertos parámetros que afectan a la aplicación y su comportamiento sin necesidad de modificar el código de la misma.

Existen diferentes archivos de configuración y dentro de ellos podremos ver distintos parámetros, cada uno de ellos con una utilidad diferenciada. A continuación se describirán dichos archivos con mayor detalle.

El primero de los archivos de configuración es el **web.xml.** Es el descriptor de despliegue de la aplicación y su función es contener toda la información necesaria para

que el contenedor web pueda desplegar una aplicación web. Este archivo en nuestra aplicación registra la siguiente información:

- Página de bienvenida de la aplicación.
- Tiempo de expiración de la sesión de usuario si está inactiva.
- Nombres y valores de los parámetros generales de la aplicación.
- Mapeo entre los Servlets y la ruta (URI) para acceder a los mismos.
- Mapeo entre las páginas de error y las excepciones de la aplicación o el código de error HTTP (Tabla 8).

Como herencia de los sistemas Pegaso y Gades, se dispone del archivo de propiedades **Configuracion.properties**. Este archivo define los nombres y valores de ciertas propiedades que serán leídos desde la aplicación y utilizados desde su código. Estas propiedades son las siguientes:

- Nombre de usuario para acceder a la base de datos.
- Contraseña de usuario para acceder a la base de datos.
- Ruta de acceso a la base de datos de Galatea.
- Ruta de acceso a la base de datos de Pegaso y Gades.
- Clase para el conector de la base de datos.

Mediante la modificación de los valores de estas propiedades podremos, por ejemplo: conectarnos a otra base de datos o utilizar otras credenciales de conexión.

El último de los archivos de configuración, **logging.properties**, establece las propiedades necesarias para configurar el registro de trazas en las bitácoras o archivos de log de la aplicación.

El contenedor web *Tomcat* dispone de una implementación propia del API *java.util.logging* que permite una configuración sencilla del registro de trazas generando un registro de bitácora por aplicación web. Esta implementación se denomina *Java Util Logging Implementation* (JULI). En este archivo de propiedades se establecen las siguientes propiedades:

- Nivel de depuración por defecto para registrar trazas.
- Directorio de almacenamiento de los archivos de registro de trazas.
- Prefijo de los archivos de registro de trazas.
- Clases para manejadores y formateadores de las trazas de registro.

El parámetro que se modifica más habitualmente es el del nivel de depuración. Este parámetro establece a partir de qué nivel de depuración se registran las trazas originadas desde la aplicación en los archivos de bitácora.

La ventaja de configurar todos estos parámetros de la aplicación en archivos de texto está en permitir, en caso de necesidad, modificar rápidamente los valores de estos parámetros en el archivo y reiniciar la aplicación para aplicar los cambios sin necesidad de modificar el código de la misma o desplegarla de nuevo.

Capítulo 4. Resultados

Una vez descrito el sistema implementado se va a proceder a exponer los resultados obtenidos en las pruebas de dicho sistema. Estas pruebas van a ser descritas agrupándolas en los siguientes procesos funcionales del sistema:

- Autenticación, inicio y cierre de sesión.
- Propuesta de una nueva pregunta.
- Propuesta de cambios sobre preguntas existentes.
- Consulta de propuestas realizadas por el usuario.

Se van a describir las pruebas más relevantes a las que se ha sometido el sistema desde el punto de vista funcional, realizando (a medida que se describen las pruebas) los comentarios interpretativos necesarios.

4.1. Autenticación: inicio y cierre de sesión

La autenticación del usuario permite iniciar una sesión en el sistema, lo que supone la habilitación del usuario para realizar el resto de tareas. Del mismo modo, se permite el cierre de dicha sesión por parte del usuario. En este apartado se describen las pruebas realizadas para verificar estas funcionalidades. Pueden autenticarse en el sistema, cualquiera de los usuarios de las plataformas Gades y Pegaso.

4.1.1. Inicio de sesión

La página de inicio del sistema Galatea incluye un formulario que permite introducir a un usuario su nombre y su contraseña (Figura 41)

Figura 41: Página de entrada al sistema Galatea

Al introducir las credenciales de usuario para iniciar sesión, si los datos no son correctos el sistema mostrará el error que corresponda, como muestra la Figura 42.

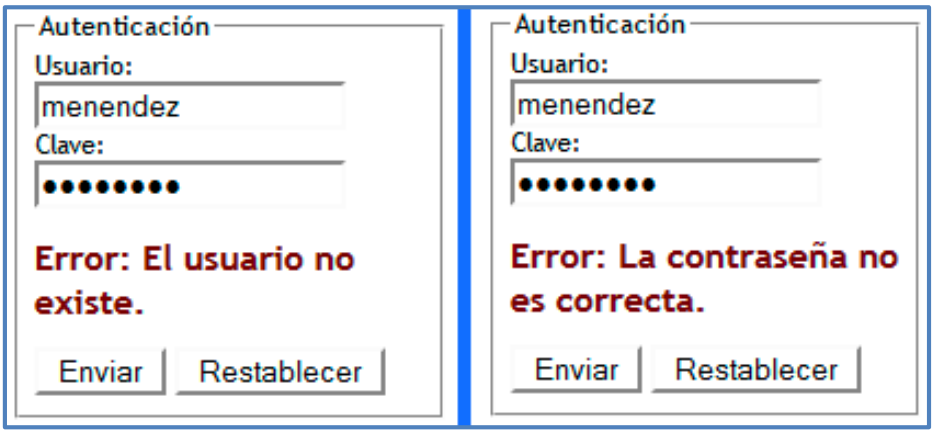


Figura 42: Errores en la autenticación del usuario

Cuando se introducen las credenciales correctas, el sistema proporciona acceso al usuario mostrándole la página de bienvenida, con acceso a las funcionalidades disponibles.

4.1.2. Cierre de sesión

Una vez autenticado el usuario en el sistema, el usuario puede elegir salir del mismo y cerrar la sesión abierta pulsando el botón “Cerrar sesión”, como muestra la

Figura 43. Este botón se muestra en cualquier página de la plataforma una vez iniciada la sesión.

Figura 43: Cabecera de Galatea con sesión iniciada

Pulsando este botón, se solicita al usuario confirmación acerca de salir del sistema. Al confirmarlo, se cierra la sesión abierta y se muestra la página de inicio.

4.2. Propuesta de una nueva pregunta

La propuesta de nuevas preguntas por parte de los usuarios de Pegaso y Gades ha sido descrita funcionalmente en la Figura 19: Diagrama de actividad "Proponer nueva pregunta". Dicha propuesta debe contener la información necesaria para una correcta evaluación posterior por parte de los expertos.

El proceso de nueva propuesta comienza al pulsar el botón "Proponer nueva pregunta" de la vista de bienvenida. La Figura 44 muestra la vista que se mostrará al usuario:

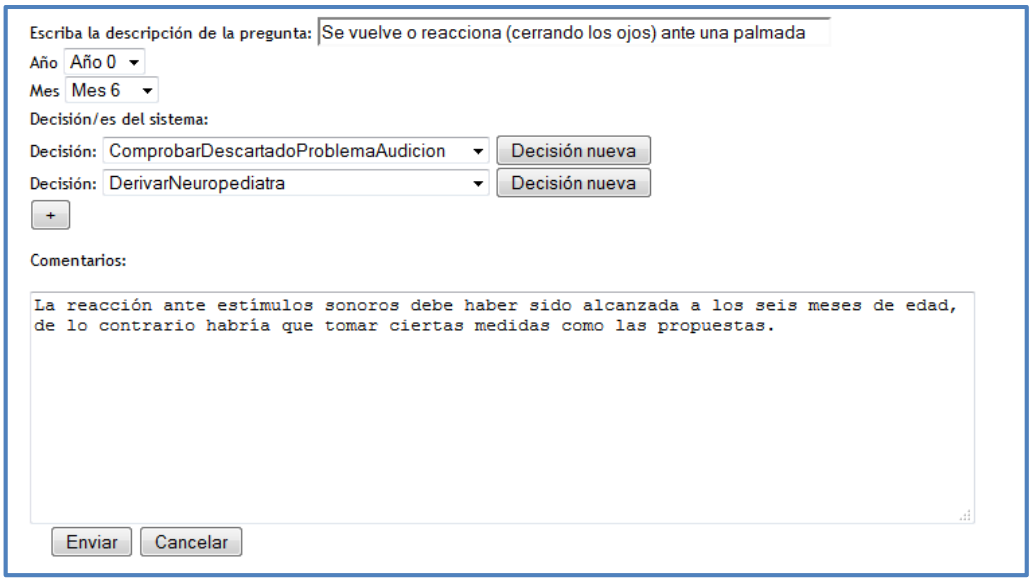


Figura 44: Captura de datos de la nueva pregunta

En la vista anterior, se rellenan los campos que componen la pregunta (descripción de la pregunta, año y mes, decisión/decisiones del sistema y comentarios). Posteriormente, se envían al servidor los datos introducidos, lo cual nos lleva a la vista

mostrada en la Figura 45, donde podremos revisar los datos de la propuesta introducida.

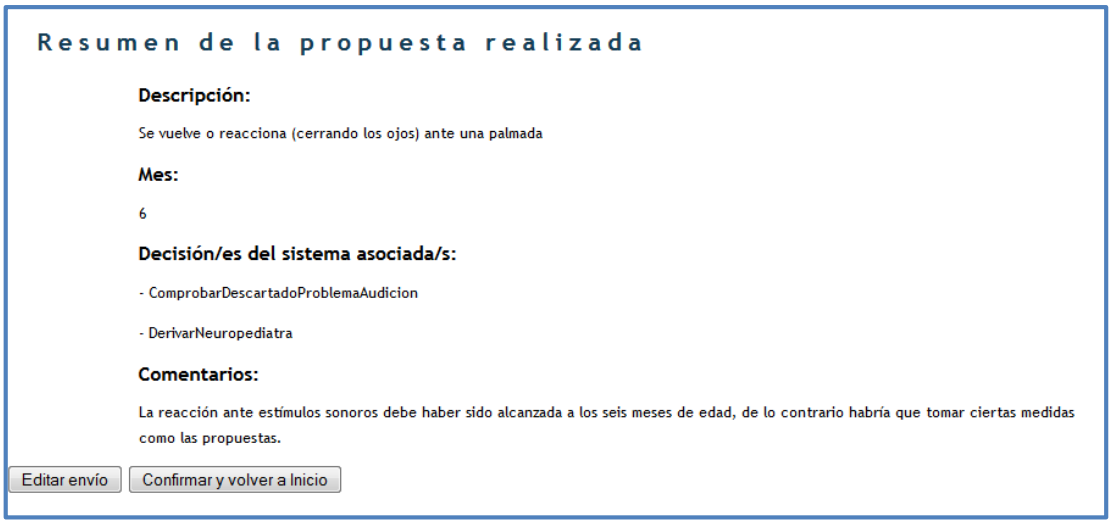


Figura 45: Revisión de la propuesta de nueva pregunta

Una vez revisada la propuesta, si se selecciona "Editar envío" se vuelve a la vista anterior con los datos completados, y en esa pantalla se podría actualizar/editar cualquiera de los datos. Por el contrario, si se consideran correctos los datos se pulsa "Confirmar y volver a Inicio", lo cual se traduce en el almacenamiento de la propuesta y se vuelve a mostrar la vista de bienvenida. Es importante señalar que una vez introducida una propuesta en el sistema, los datos de esta propuesta no pueden ser modificados.

4.3. Propuesta de cambios sobre preguntas existentes

La propuesta de cambios sobre las preguntas contenidas en la base de conocimiento de las plataformas Pegaso y Gades ha sido descrita en la Figura 20: Diagrama de actividad "Proponer cambios en preguntas existentes".

Este proceso de propuesta de cambios comienza al pulsar el botón "Proponer cambios" de la vista de bienvenida. La Figura 46 muestra la vista que verá el usuario a continuación, donde se elegirá en primer lugar el año y los meses a los cuales pertenecen las preguntas:

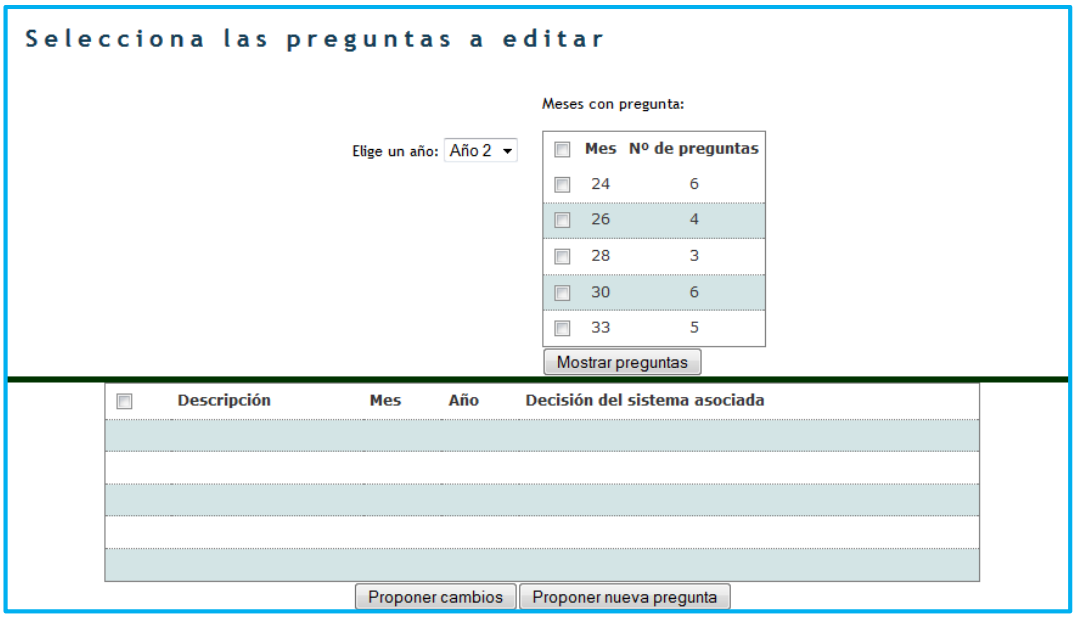


Figura 46: Selección de año y meses de las preguntas a editar

Una vez elegido el/los mes/es de las preguntas, se mostrarán en la tabla inferior las preguntas que pertenecen a dichos meses, como se muestra en la siguiente figura:

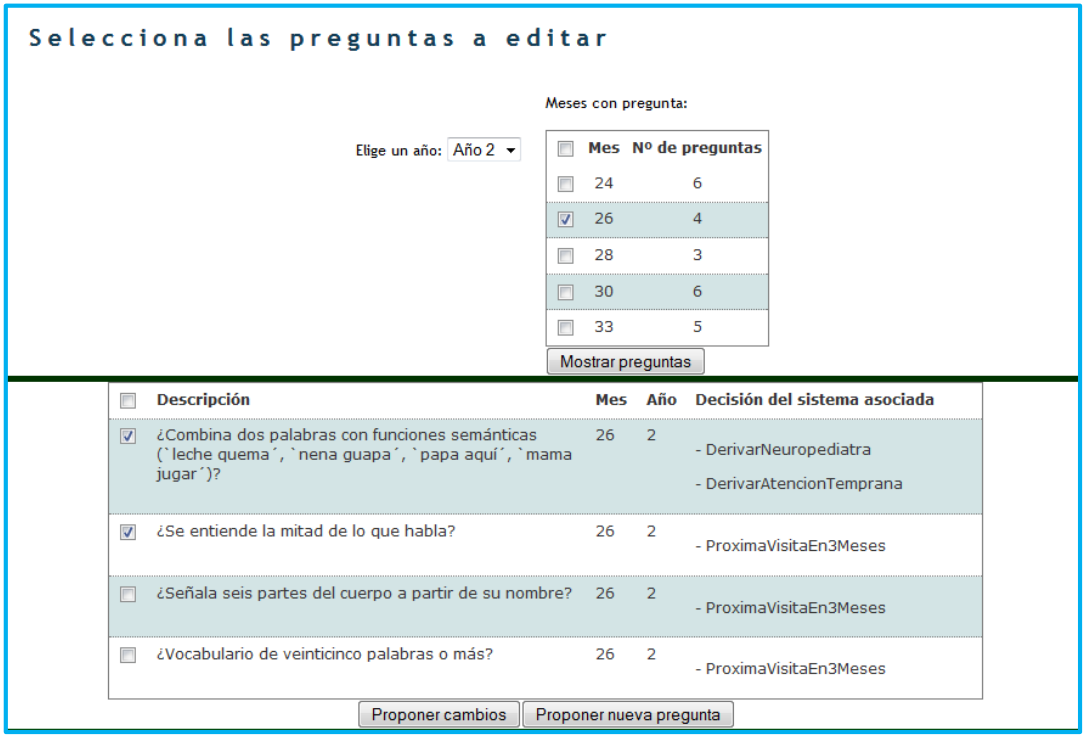


Figura 47: Selección de preguntas a editar

La figura anterior permite introducir propuestas de cambios sobre las preguntas del mes 26 (segundo año).

Al observar las preguntas tomamos la decisión de proponer cambios sobre dos de ellas, por lo que las señalamos y pulsamos "Proponer cambios". Lo que nos lleva a la pantalla de edición, ilustrada mediante la Figura 48.

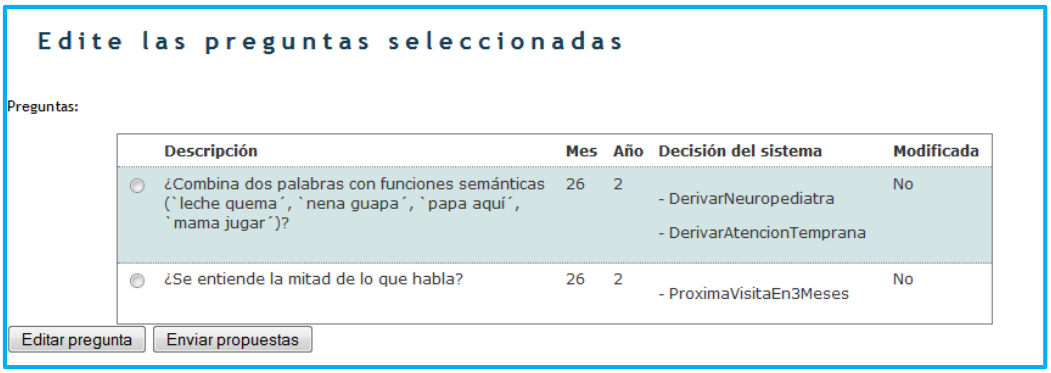


Figura 48: Editor de preguntas

A continuación, procedemos a editar una pregunta, tras observar el resumen completo y mediante las facilidades que se muestran (Figura 49).

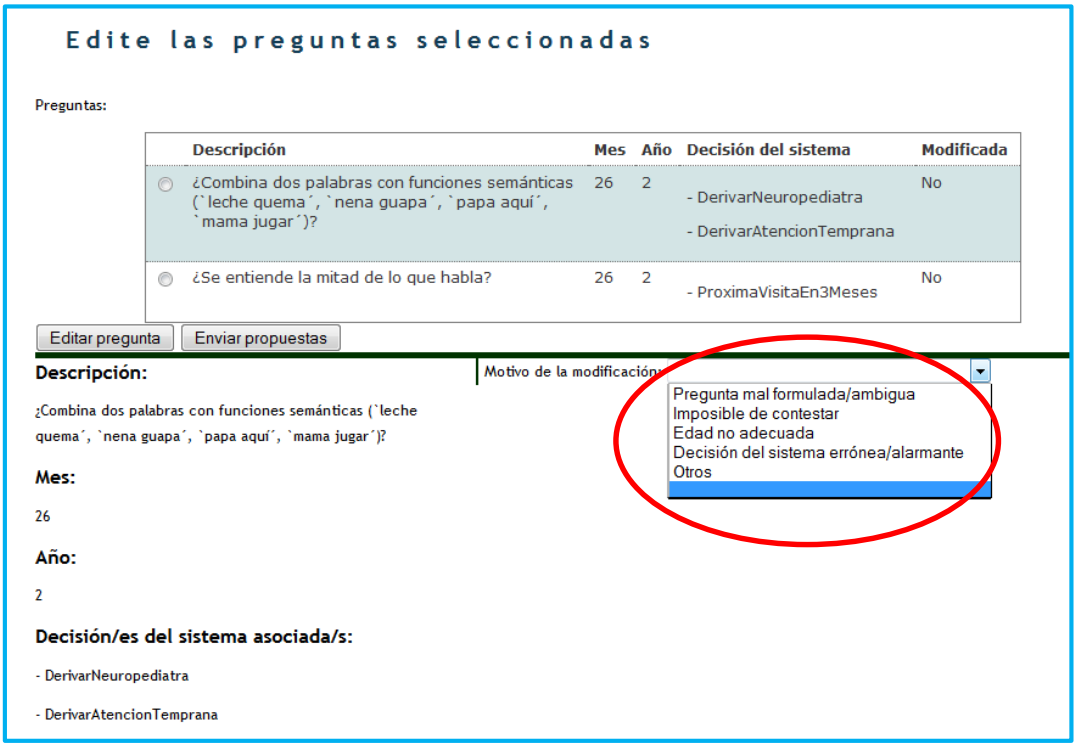


Figura 49: Selección del motivo de la propuesta de cambio

Tras la elección de un motivo se nos muestran los campos necesarios para realizar la propuesta, los diferentes motivos se muestran en la figura anterior.

A continuación, se mostrarán los campos que aparecen según el motivo elegido:

- Propuesta mal formulada/ambigua.

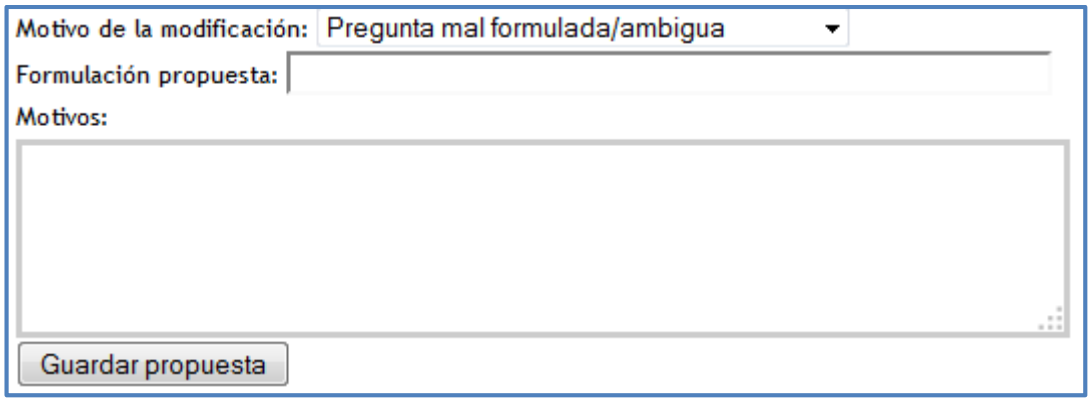


Figura 50: Formulario para los cambios por error en la formulación de la pregunta

- Imposible de contestar.

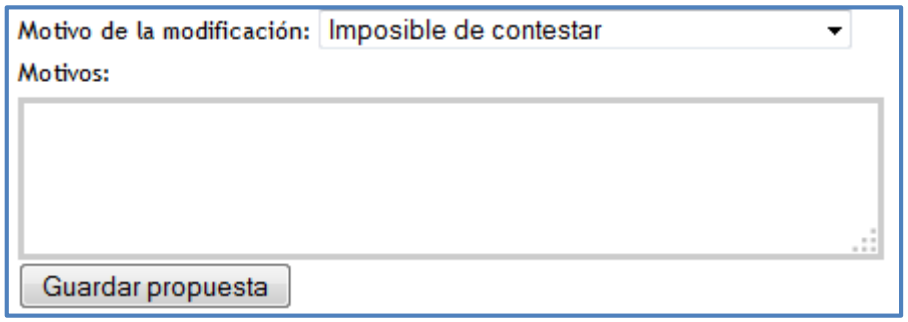


Figura 51: Formulario para los cambios por imposibilidad de contestar la pregunta

- Edad no adecuada.

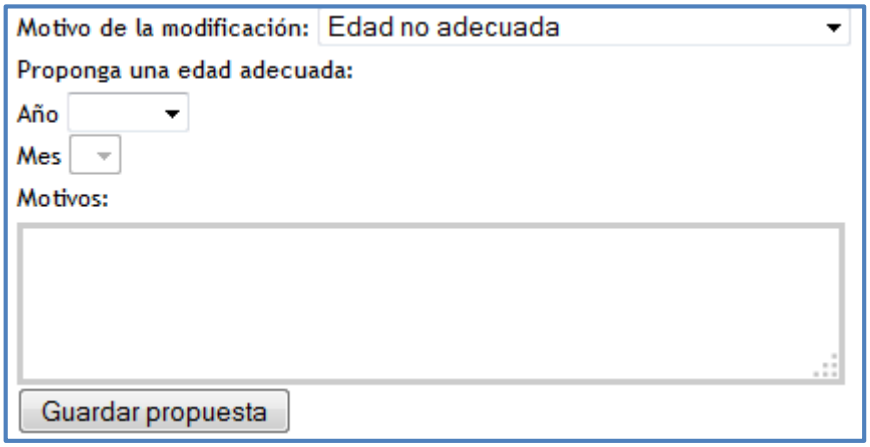


Figura 52: Formulario para los cambios por edad no adecuada

- Decisión del sistema errónea/alarmante. La siguiente figura muestra lo que se debe introducir en este caso. Es importante señalar que si esto ocurre se pueden introducir nuevas decisiones del sistema para esta pregunta, dando al usuario la posibilidad de introducir más de una decisión del sistema por pregunta.

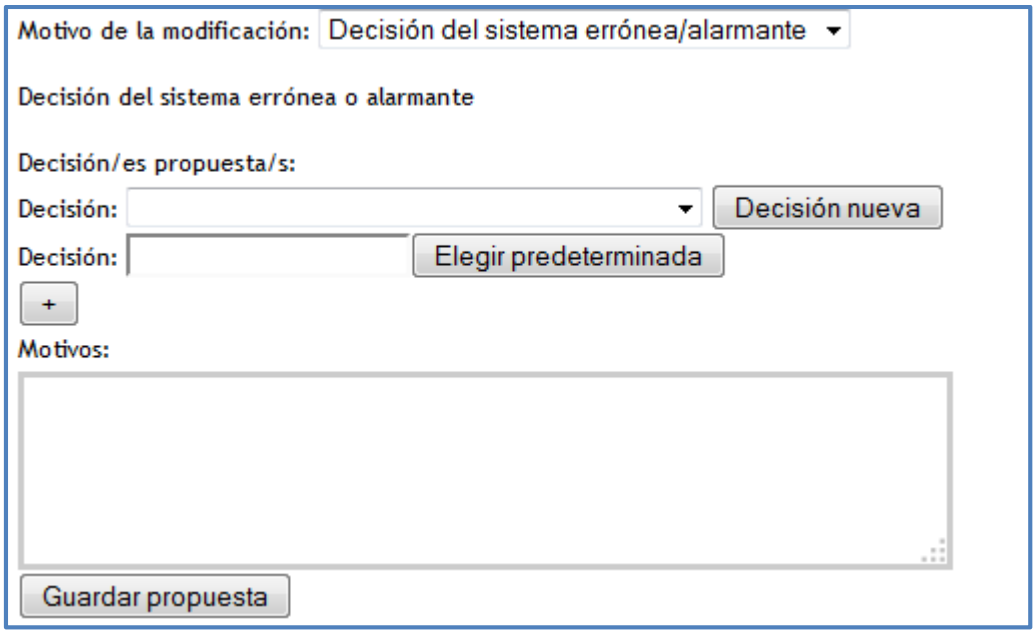


Figura 53: Formulario para los cambios por decisiones del sistema erróneas

- Otros.

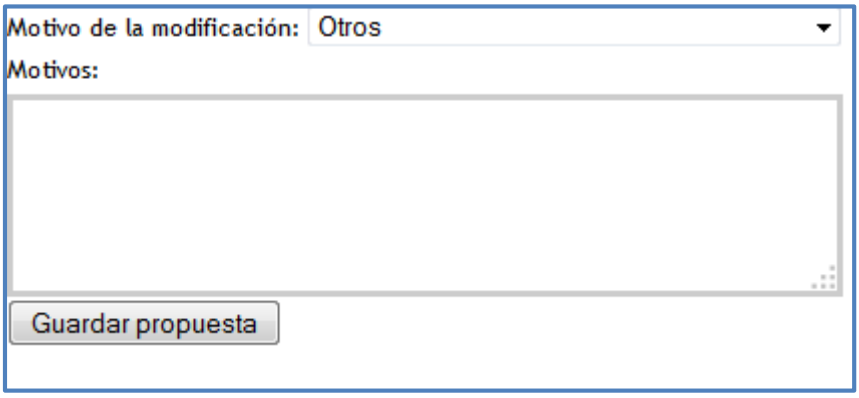


Figura 54: Formulario para los cambios por otros motivos

Tras elegir el motivo y completar los campos necesarios, la tabla principal que muestra el editor se actualiza, permitiéndonos observar los cambios (si hay cambios visibles) y proceder a editar el resto (Figura 55).

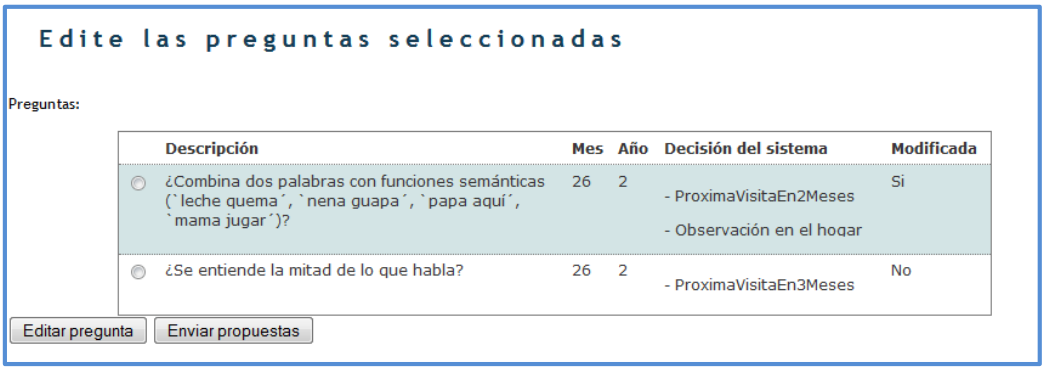


Figura 55: Editor de preguntas tras realizar una propuesta

Tras editar todas, se envían las propuestas, lo que nos lleva a la vista resumen para evaluar nuestras propuestas y, de ser correctas confirmar su envío y volver al Inicio (Figura 56).

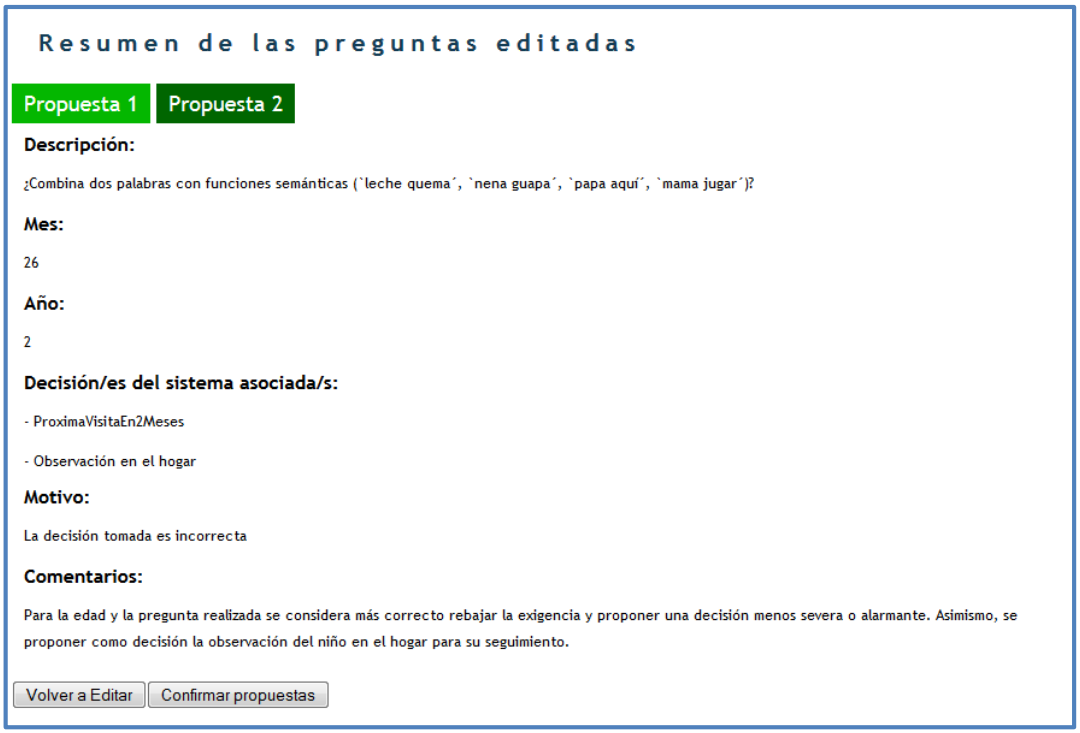


Figura 56: Vista de resumen para la confirmación de propuestas de cambio

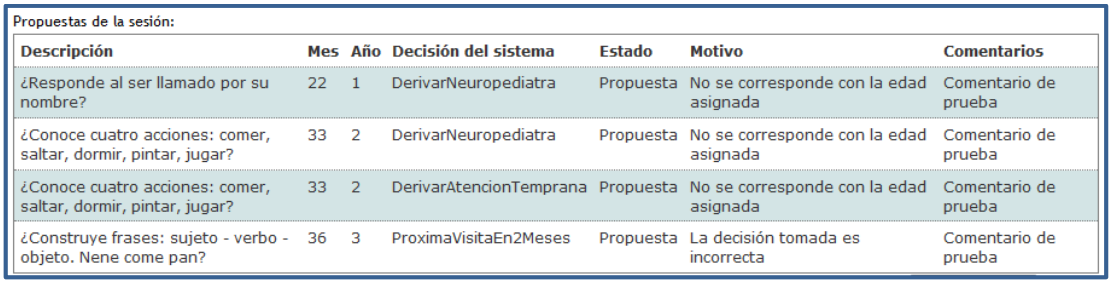
4.4. Consulta de propuestas realizadas por el usuario

El proceso de consulta de las propuestas realizadas por el usuario comienza al escoger esta funcionalidad en la página de bienvenida. Tras ello, se mostrará al usuario la vista de consulta (Figura 57).

Figura 57: Consulta de propuestas realizadas

En esta pantalla, el usuario puede elegir entre consultar las propuestas realizadas en la sesión actual (desde que inició sesión autenticándose) o todas las propuestas realizadas utilizando la plataforma Galatea.

Si elige mostrar las propuestas de la sesión actual, observará una tabla como la que ilustra la Figura 58.



Propuestas de la sesión:

Descripción	Mes	Año	Decisión del sistema	Estado	Motivo	Comentarios
¿Responde al ser llamado por su nombre?	22	1	DerivarNeuropediatra	Propuesta	No se corresponde con la edad asignada	Comentario de prueba
¿Conoce cuatro acciones: comer, saltar, dormir, pintar, jugar?	33	2	DerivarNeuropediatra	Propuesta	No se corresponde con la edad asignada	Comentario de prueba
¿Conoce cuatro acciones: comer, saltar, dormir, pintar, jugar?	33	2	DerivarAtencionTemprana	Propuesta	No se corresponde con la edad asignada	Comentario de prueba
¿Construye frases: sujeto - verbo - objeto. Nene come pan?	36	3	ProximaVisitaEn2Meses	Propuesta	La decisión tomada es incorrecta	Comentario de prueba

Figura 58: Consulta de propuestas realizadas en la sesión

Por el contrario, si elige mostrar el histórico completo de propuestas, esta vista incluirá mecanismos de paginación, ya que el volumen de propuestas es presumiblemente superior al de la sesión (Figura 59).

Histórico de propuestas realizadas

Mostrar propuestas de esta sesión Mostrar todas las propuestas realizadas Actualizar

Histórico de propuestas:

Descripción	Mes	Año	Decisión del sistema	Estado	Motivo	Comentarios
¿Sabe hacer la correspondencia fonema grafema de varias palabras?	74	6	DerivarNeuropediatra	Propuesta	No se corresponde con la edad asignada	Comentario de prueba
¿Es capaz de interpretar dobles intenciones ? P.ej: juegos, trampas, bromas	72	6	DerivarAtencionTemprana	Propuesta	Pregunta mal formulada/ambigua	Comentario de prueba
¿Es capaz de interpretar dobles intenciones ? P.ej: juegos, trampas, bromas	72	6	DerivarNeuropediatra	Propuesta	Pregunta mal formulada/ambigua	Comentario de prueba
¿Reconoce metáforas: rubia como el oro?	72	6	ProximaVisitaEn3Meses	Propuesta	La decisión tomada es incorrecta	Comentario de prueba
¿Cuenta hasta diez?	61	5	DerivarNeuropediatra	Propuesta	Nueva propuesta	Comentario de prueba
¿Emplea `no´ en su lenguaje?	24	2	DerivarAtencionTemprana	Propuesta	Otros	Comentario de prueba
¿Emplea `no´ en su lenguaje?	24	2	DerivarNeuropediatra	Propuesta	Otros	Comentario de prueba
¿Responde al ser llamado por su nombre?	22	1	DerivarAtencionTemprana	Propuesta	No se corresponde con la edad asignada	Comentario de prueba

1ª página... página anterior... ...página siguiente ... última página Volver a Inicio

pag 2/3

Figura 59: Consulta del histórico de propuestas

En cualquier momento el usuario puede elegir volver a la página de bienvenida pulsando sobre el botón “Volver al Inicio”, como se muestra en las figuras anteriores.

Capítulo 5. Conclusiones

El desarrollo de las tecnologías de la información y la comunicación junto con la expansión de su aplicación en el área de la salud lleva años facilitando la construcción de herramientas y aplicaciones que facilitan la vida a los profesionales del ámbito sanitario y a los pacientes que se benefician de los servicios proporcionados por la tecnología.

El presente proyecto parte del aprovechamiento de las nuevas tecnologías para el desarrollo de una herramienta que facilita la evolución supervisada del conocimiento almacenado en una aplicación ya existente. Con esta evolución, se pretende dar el primer paso en la construcción de una herramienta de consenso entre expertos necesaria para lograr el refinamiento de la Base de Conocimiento, núcleo de una aplicación de soporte a la toma de decisiones en atención primaria y escuelas infantiles.

El presente proyecto cubre la primera etapa, clave, para lograr esta evolución, la recogida de propuestas de cambio sobre la Base de Conocimiento existente. Las funcionalidades principales del sistema presentado en este proyecto, se han mostrado suficientes para cubrir las necesidades de los usuarios del sistema Galatea.

Contemplando la solución realizada, como parte de un sistema mayor, que irá evolucionando y creciendo en el tiempo, es necesario que dicho sistema esté estructurado internamente de una manera que facilite la evolución del sistema.

La utilización de tecnologías web y, más concretamente, de los patrones de modularidad y reutilización sigue esta línea de facilitar la evolución del sistema. Teniendo en cuenta, además, que se parte del conocimiento de la existencia de un desarrollo posterior que ampliará el sistema actual, así como las funcionalidades que se le pretenden. De este modo, esta solución intenta ir un paso más allá de las necesidades del propio sistema, intentado prever y cubrir necesidades posteriores que aparecerán en el desarrollo de este sistema mayor.

En el desarrollo del sistema han colaborado usuarios finales del sistema, garantizando su usabilidad y validez mediante su inclusión en la fase de pruebas.

Se ha realizado un estudio de sistemas tradicionales de consenso, que han permitido, identificar requisitos del sistema Galatea, así como posibles interacciones entre los usuarios involucrados. Este estudio permite justificar la elección del

mecanismo de consenso entre expertos, como la mejor forma de conseguir la evolución de la Base de Conocimiento de los sistemas de partida.

También se ha estudiado, en profundidad, las librerías del API Apache Jena, para lograr obtener una visión general de las funcionalidades de dicho API. El estudio fue el punto de partida para lograr generar satisfactoriamente un algoritmo que permite la generación automática del fichero OWL a partir la información almacenada en la Base de Datos del sistema Galatea.

Como conclusión final, el producto desarrollado es una herramienta con suficiente calidad y funcionalidad como para cubrir las necesidades que se plantearon al comienzo de este proyecto, otorgando a los actores implicados una plataforma que resuelva sus necesidades sin establecer nuevos problemas.

Como el tiempo para el desarrollo del proyecto es limitado, existen características por implementar que podrían hacer de éste un mejor producto. Por ello, el siguiente apartado está dedicado a las posibles mejoras o trabajos que se pueden realizar en el sistema, siempre con la vista en el conjunto de sistemas Pegaso, Gades y este mismo, Galatea, y en esa plataforma que juntos componen.

Capítulo 6. Trabajo futuro

A continuación, se enumeran algunas futuras mejoras y trabajos relacionados que pueden realizarse:

- Desarrollar un sistema basado en CSCW para la toma de decisión entre expertos, que permita la evaluación colaborativa de las propuestas y su modificación en la base de conocimiento de Pegaso y Gades.
- Considerar la posibilidad de editar las propuestas realizadas, ya que en el sistema desarrollado no existe esta posibilidad. Son aspectos a tener en cuenta la posterior modificación por parte de los expertos o el convencimiento que se presupone al usuario al hacer dichas propuestas.
- Proporcionar plataformas diferenciadas a los usuarios de Pegaso y Gades que permitan realizar propuestas de cambio sobre las bases de conocimiento de sus respectivas plataformas, debido a que Galatea parte de una de las Bases de Conocimiento, la de Gades o Pegaso, para realizar propuestas.
- Proporcionar a los usuarios de Galatea de un conjunto de posibles propuestas de cambio en función de resultados estadísticos. Es decir, preguntas que siempre tienen como respuesta: SI, NO o NS/NC, pueden necesitar ser revisadas por el grupo de expertos.
- Valorar la utilidad de generar formularios que puedan imprimir resultados de propuestas de cambio.
- Facilitar, a los usuarios de Galatea, formularios en papel que permitan hacer una recogida de posibles propuestas de cambio. A partir de la información recopilada en estos formularios, el usuario puede tardar menos tiempo en introducir las propuestas en Galatea.

Capítulo 7. Referencias

[ALU01] "*J2EE Patterns*". Deepak ALUR, John CRUPI, Dan MALKS, (2001) Sun Java Center, Oracle.com.

[IND08] "*Collaborative Decision Making framework for Multi-Agent System*". Indiramma, M. y Anandakumar K.R. Research Scholar, PET Research Foundation, PESCE y Profesor and HOD, Dept of CSE, SJBIT.

[KAR99] **"***Computer supported argumentation and collaborative decision making: the Hermes system.*" Karacapilidis, N. y Papadias, D. Industrial Management Laboratory, University of Patras, Greece y Department of Computer Science, Hong Kong University of Science and Technology, Clearwater Bay, Hong Kong.

[MAK98] **"***Teleworks: A CSCW Application for Remote Medical Diagnosis Support and Teleconsultation.*" Makris, L.; Kamilatos, I.; Kopsacheilis, E. V.; y Strintzis, M. G. IEEE.

[MAR13] "*Arquitectura Telemática para la Detección Precoz de Trastornos del Lenguaje*". Martín Ruiz, M.L.; Valero Duboy, M.A.; Torcal Loriente, C.; Martín Uría, J.; Peñafiel Puerto, M. Madrid : s.n., 2013.

[PAL14] **"***Análisis de datos y modelado simbólico para aplicación de detección temprana de trastornos del lenguaje*". Palomo Díaz, Ó. Madrid: ETS de Ingeniería y Sistemas de Telecomunicación - UPM, 2014.

[URI13] "*Implementación de una arquitectura de componentes para un sistema de apoyo*". Martín Uría, J. Madrid: ETS de Ingeniería y Sistemas de Telecomunicación - UPM, 2013.

[WIK14] Wikipedia: la enciclopedia libre.

http://en.wikipedia.org/wiki/Collaborative_decision-making_software

http://en.wikipedia.org/wiki/Group_decision-making

http://en.wikipedia.org/wiki/Web_Ontology_Language

http://en.wikipedia.org/wiki/Model_view_controller

Consultado en julio de 2014.

Anexo I: Manual de usuario del sistema Galatea

Plataforma de conocimiento evolutivo para detección temprana de trastornos del lenguaje

Galatea

Manual de usuario

1. Entrar al sistema

La página de entrada al sistema tiene el siguiente aspecto:

Fig. 1: Página de entrada al sistema Galatea

Esta página permite, a los usuarios de los sistemas Pegaso y Gades, iniciar sesión en el sistema Galatea, mediante la introducción del usuario y contraseña que ya poseen de dichas plataformas.

Estos usuarios pueden iniciar sesión en el sistema introduciendo su nombre de usuario y contraseña en los campos que se indican en la y pulsando la tecla **Enter** o bien el botón **Enviar.**

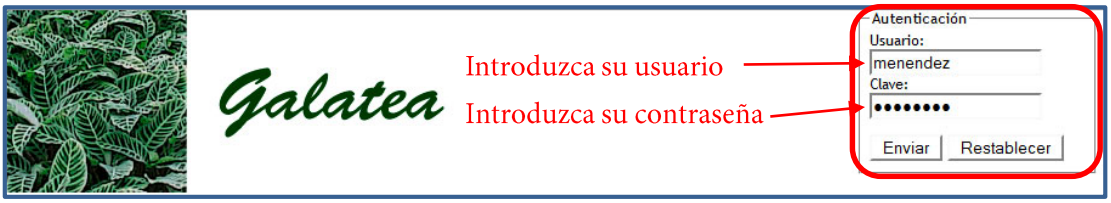


Fig. 2: Formulario de autenticación en Galatea

Si se introduce un usuario que no existe o una contraseña errónea se mostrarán los siguientes mensajes de error, respectivamente:

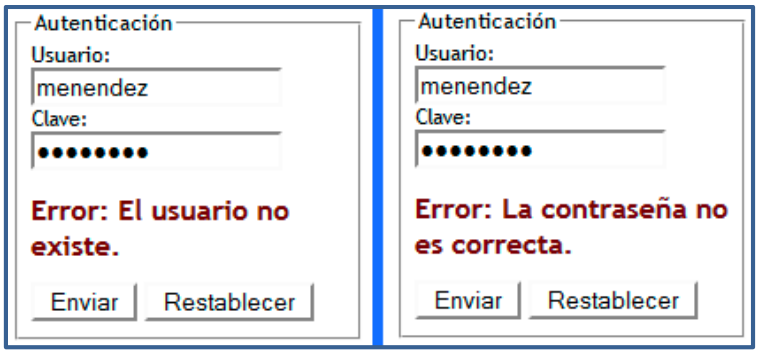


Fig. 3: Errores en la autenticación del usuario

Si observa estos errores vuelva a intentarlo prestando atención a la información introducida y recordando que el sistema distingue minúsculas y mayúsculas en ambas credenciales del usuario. Si el problema persiste o no recuerda su usuario o contraseña, póngase en contacto con el administrador del sistema para que se las proporcione.

Una vez introducidas las credenciales correctas se le dará acceso a la página de bienvenida. Observará en el lugar donde se encontraba el formulario que aparecerá su nombre de usuario. Tendrá acceso a las funcionalidades que se muestran señaladas en la Fig. 4: Proponer nueva pregunta, Proponer cambios y Consultar el histórico de propuestas.

Fig. 4: Página principal con funcionalidades habilitadas

2. Propuesta de nueva pregunta

Si durante la utilización de las plataformas Pegaso o Gades (según sea su caso), detecta preguntas que podría realizar el sistema para evaluar más correctamente el desarrollo del lenguaje de los niños, esta funcionalidad le permite transmitir esta información al sistema. Si utiliza el acceso mostrado en la Fig. 4, será guiado por una serie de páginas donde podrá realizar de forma asistida dicha propuesta de nueva pregunta.

La primera de las páginas que visualizará (Fig. 5) le permitirá introducir todos los datos de una pregunta: descripción de la pregunta; año y mes en los que se formulará la pregunta y, por lo tanto, debe haber sido superado el hito de adquisición del lenguaje por el niño evaluado; decisión o decisiones del sistema asociadas a la pregunta, es decir, las recomendaciones del sistema ante una respuesta no satisfactoria a la pregunta; y los comentarios, los cuales contienen la motivación de la propuesta, su justificación.

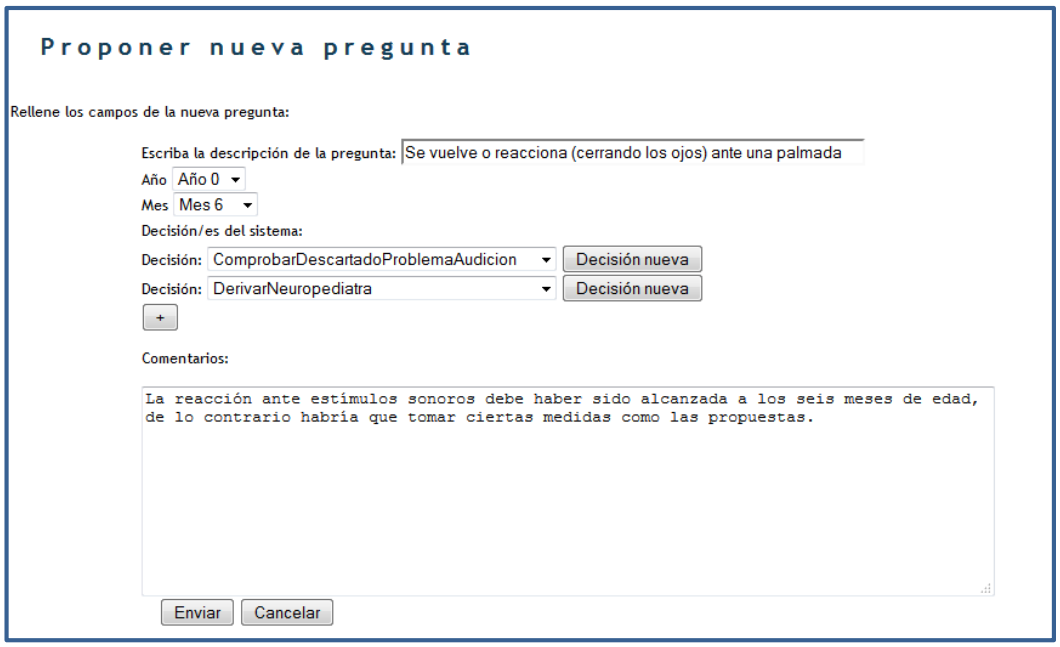


Fig. 5: Formulario para la introducción de una nueva propuesta de pregunta

En este punto, el usuario puede asociar un número variable de decisiones del sistema. Por defecto, como se observa en la figura anterior, aparece un campo de decisión, pero si pulsamos sobre el botón "+", aparecerá un nuevo campo que nos permite seleccionar otra decisión.

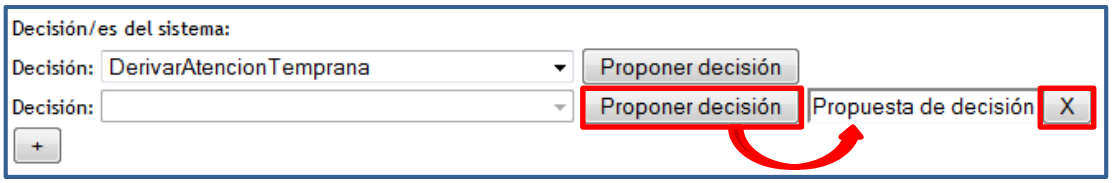


Fig. 6: Decisiones asociadas a una propuesta

Del mismo modo, puede considerar que las opciones predefinidas como decisiones del sistema no se ajustan a sus necesidades, por lo que tiene la opción de, pulsando sobre el botón "Proponer decisión", introducir manualmente una propuesta de decisión del sistema para esta pregunta.

Si se decide finalmente elegir una decisión predeterminada, se puede revocar la decisión anterior pulsando el botón que muestra la figura anterior ("X").

Una vez rellenados los campos de la pregunta y elegir enviar la propuesta al sistema, éste le mostrará una vista de resumen de la propuesta realizada con el fin de revisarla, tal y como se muestra en la Fig. 7.

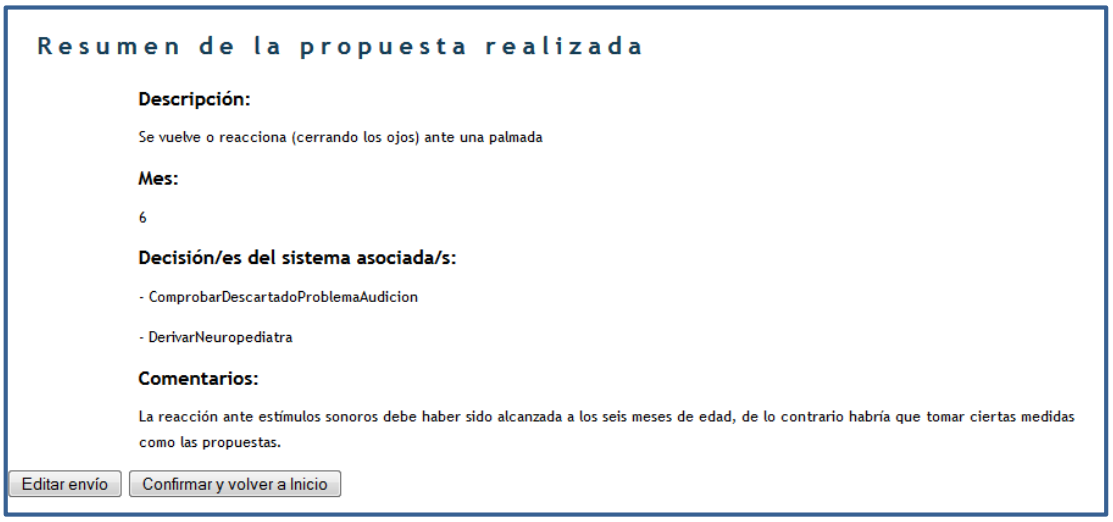


Fig. 7: Vista de resumen de la propuesta de nueva pregunta

Esta revisión le permite observar la propuesta de una manera ordenada y considerar si desea volver a editarla (Botón "Volver a Editar) o, si es correcta, confirmar su envío definitivo al sistema para su almacenamiento (Botón "Confirmar y enviar"). Cabe destacar que el sistema no le permitirá modificar las propuestas una vez confirmado su envío, de ahí que exista la vista de resumen que le permite revisarla previamente.

3. Propuestas de cambios sobre preguntas existentes

Como ocurre con las nuevas preguntas, usted puede observar errores o fallos que, bajo su consideración, debieran ser corregidos para el buen funcionamiento de las evaluaciones de los sistemas Pegaso y Gades. Si utiliza el acceso mostrado en la Fig. 4, será guiado por una serie de páginas donde podrá realizar de forma asistida dichas propuestas de cambios.

Al elegir "Proponer cambios" en la página de bienvenida, se mostrará la siguiente vista (Fig. 8):

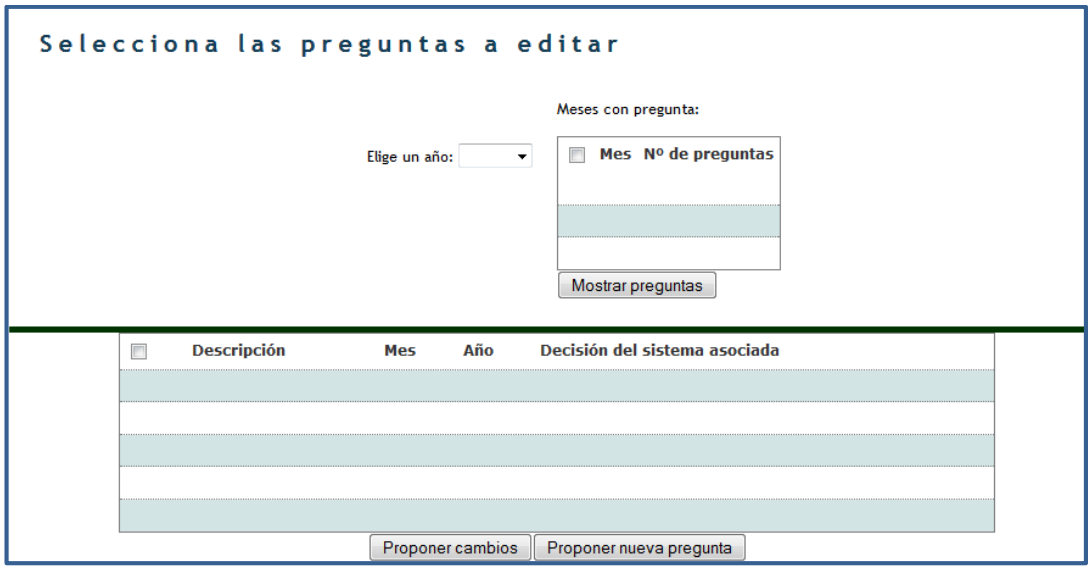


Fig. 8: Página de selección de preguntas a editar

En primer lugar deberá introducir el año al que pertenecen las preguntas sobre las que desea realizar propuestas de cambio, mediante el menú desplegable señalado en la figura anterior.

Al elegir el año, verá en la tabla anexa como aparecen los meses de dicho año de los que existen preguntas y cuántas son. Esta tabla le permite elegir los meses de los que quiere mostrar las preguntas que se realizan. Si quiere que se muestren las preguntas de todos los meses, el botón de la cabecera de la tabla le permite seleccionarlos todos.

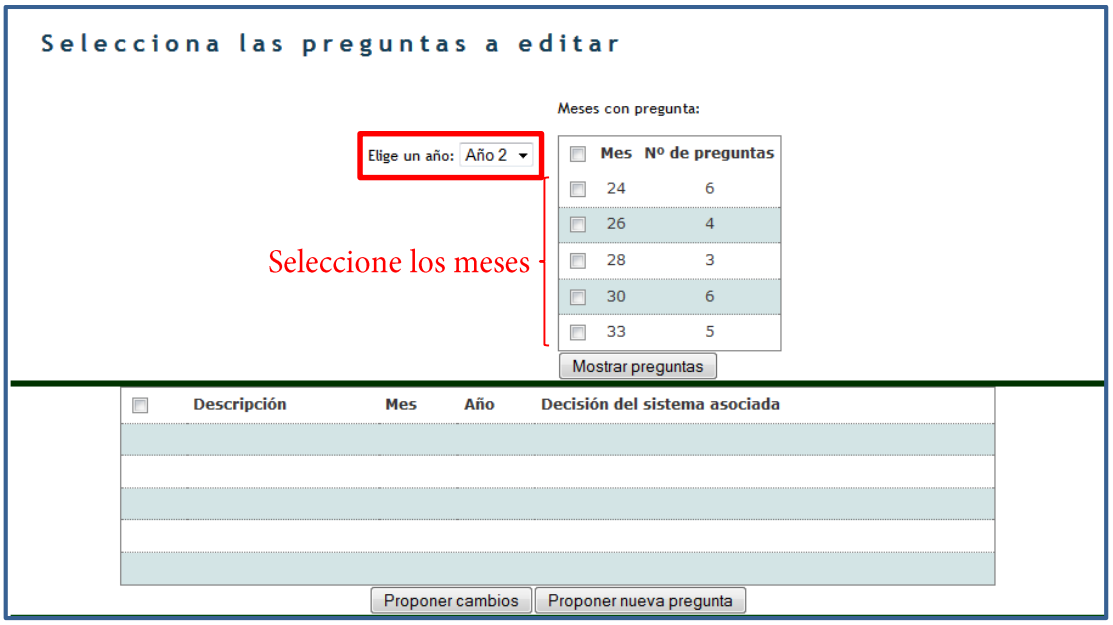


Fig. 9: Selección de año y meses de las preguntas a editar

Una vez elegidos los meses de los que mostrar las preguntas, pulsando el botón "Mostrar preguntas", le serán mostradas todas las preguntas que cumplan lo anterior. Del mismo modo que en la tabla anterior, podrá seleccionar todas las preguntas sobre las que desee realizar propuestas. Como muestra la Fig. 10, se ofrecen dos opciones: "Proponer cambios" sobre las preguntas seleccionadas, o bien "Proponer nueva pregunta", lo que le llevaría al proceso descrito en el apartado anterior.

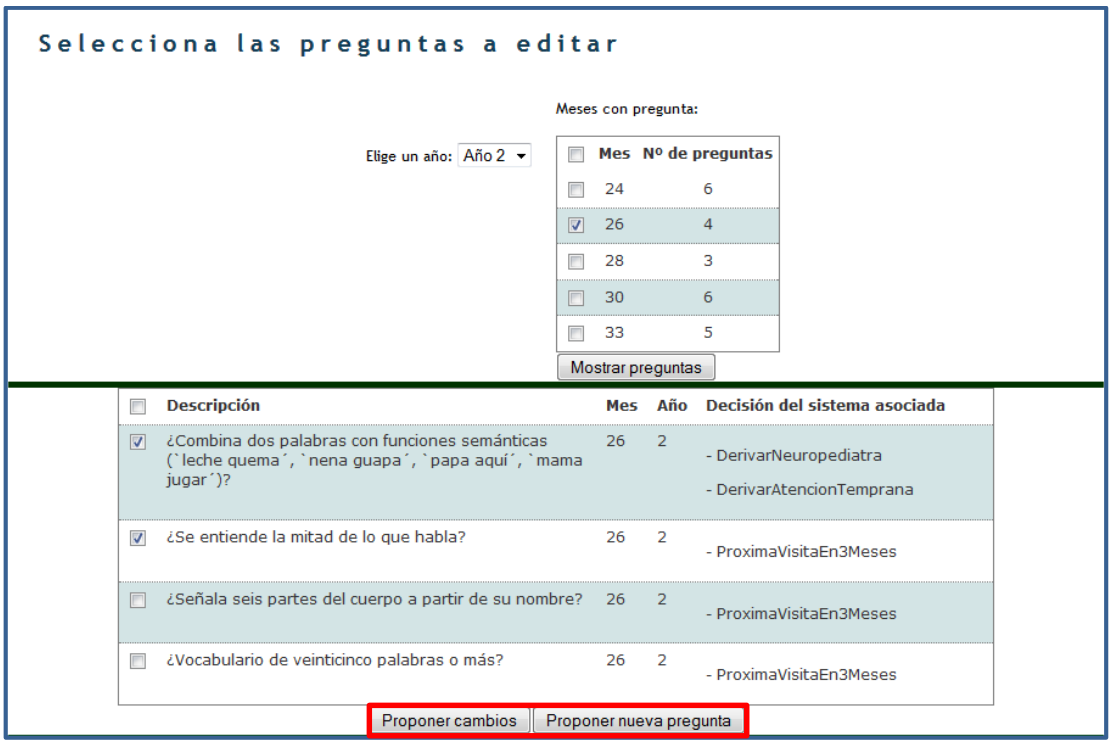


Fig. 10: Vista de las preguntas correspondientes al año y meses seleccionados

Si decide proponer cambios sobre las preguntas seleccionadas, el sistema le mostrará la página de edición, donde se editarán las preguntas una a una (Fig. 11).

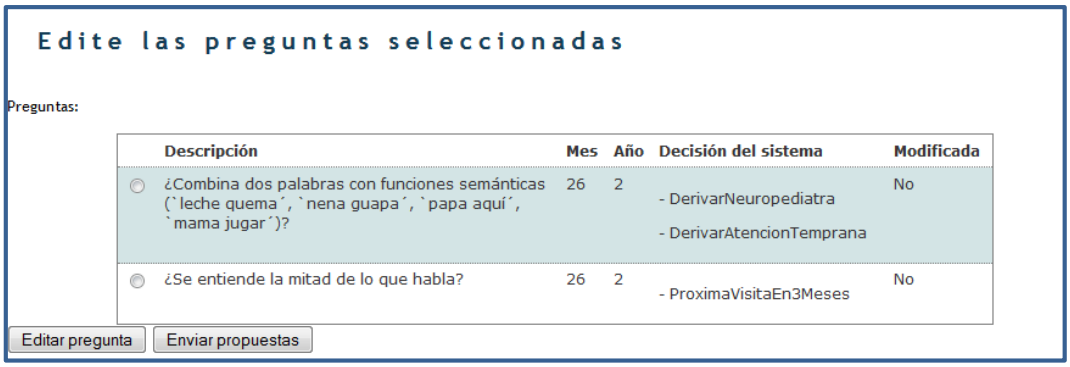


Fig. 11: Editor de preguntas

La tabla anterior le muestra una descripción de la pregunta, así como si sobre ella se ha realizado una propuesta (Campo "Modificada"). Al elegir una pregunta y pulsar sobre "Editar pregunta", se muestra un nuevo apartado (Fig. 12).

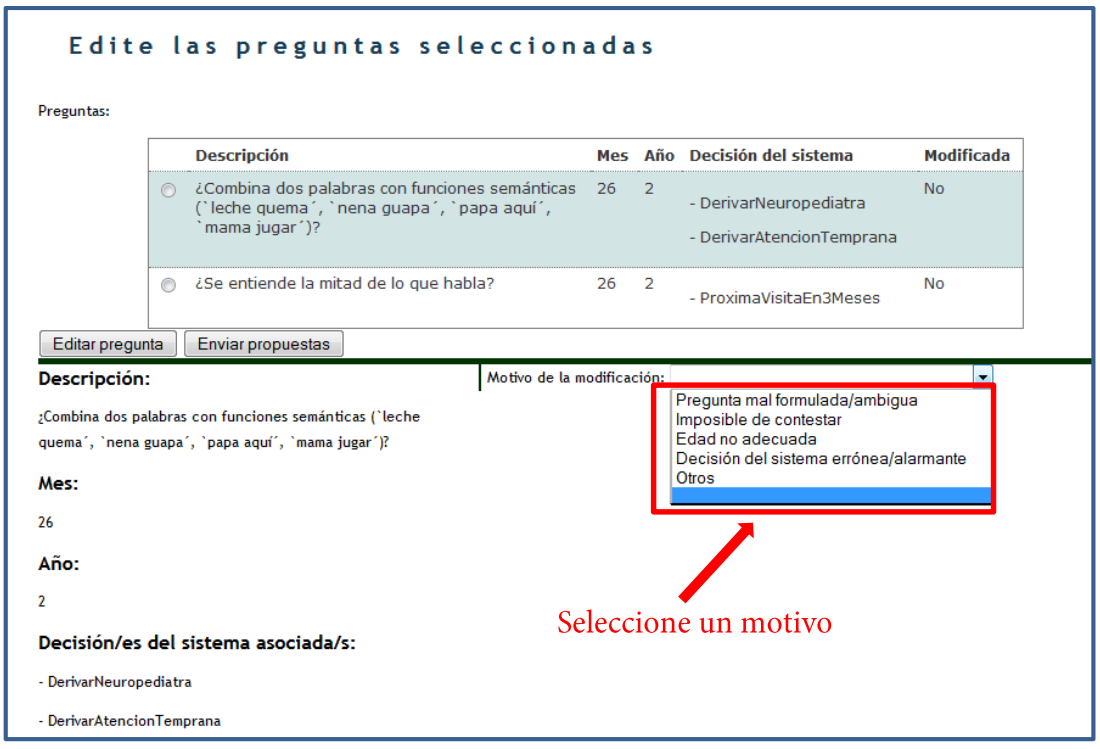


Fig. 12: Proceso de edición de una pregunta

En él, puede elegir el motivo de su propuesta de cambio, según el cual se le ofrecerán los campos necesarios para realizarla. Estos motivos son:

A. Pregunta mal formulada/ambigua

   Lo cual le ofrecerá la posibilidad de reformular la pregunta para resolver el conflicto, así como de justificar esta propuesta de cambio (Fig. 13).

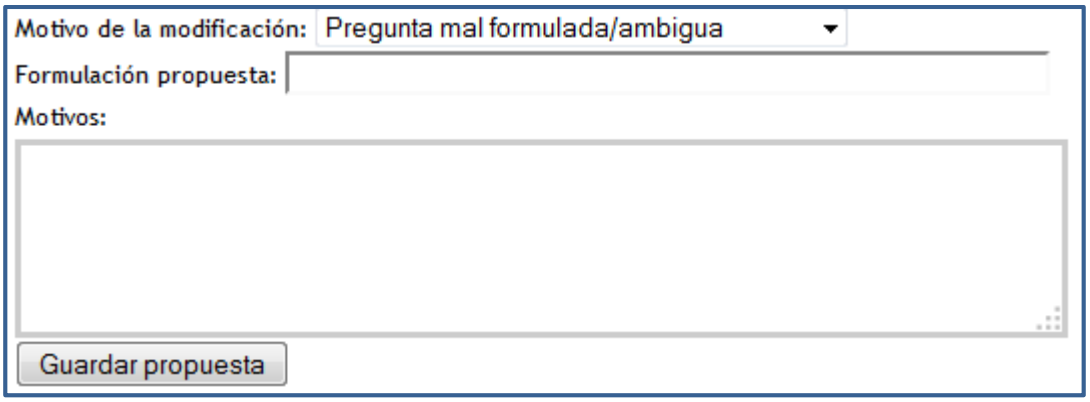


Fig. 13: Formulario pregunta mal formulada

B. Imposible de contestar

Este motivo le ofrecerá la posibilidad de indicar mediante los comentarios que le impide contestar a dicha pregunta (Fig. 14).

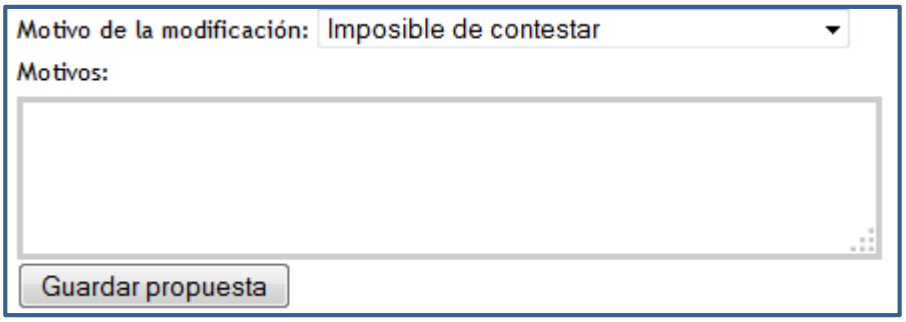


Fig. 14: Formulario pregunta imposible de contestar

C. Edad no adecuada

Esta opción le permitirá escoger la edad en meses que considere adecuada para la pregunta, así como una justificación de los motivos de esta propuesta.

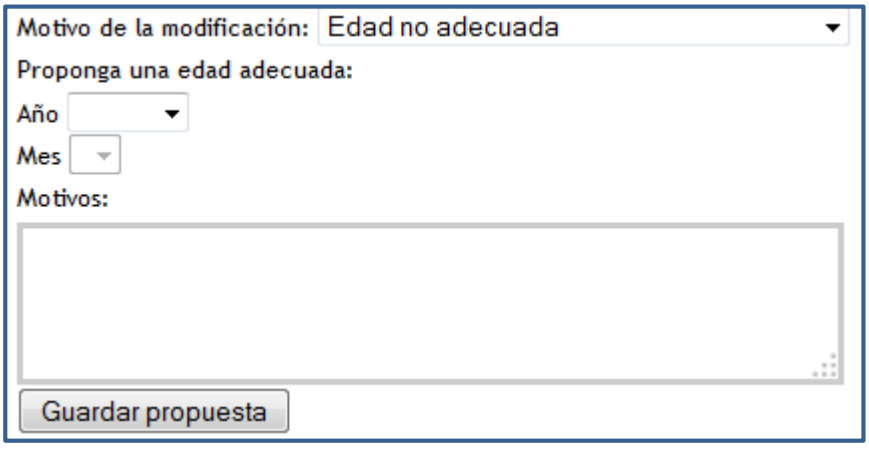


Fig. 15: Formulario edad de la pregunta no adecuada

D. Decisión del sistema errónea/alarmante

Si considera que la propuesta obedece a este motivo, el sistema le mostrará un pequeño formulario como el mostrado en la Fig. 16, que le permita elegir una o varias decisiones predeterminadas o propuestas por usted (como ocurría para "Proponer nueva pregunta"). Como anteriormente, la propuesta debe justificarse mediante los motivos.

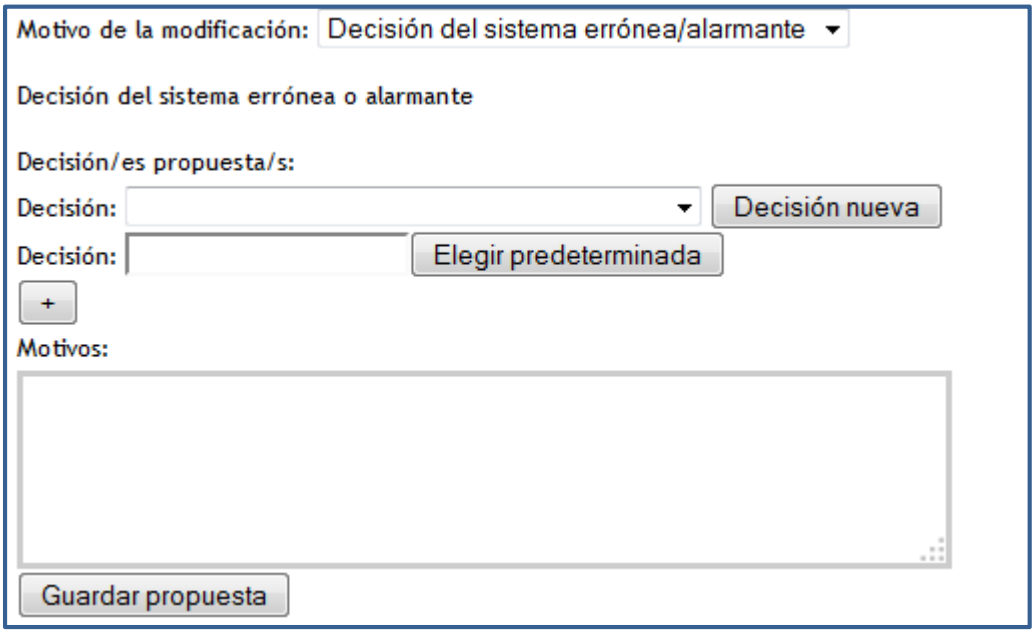


Fig. 16: Formulario decisión errónea o alarmante

E. Otros

Si su propuesta no obedece a ninguno de los motivos anteriores, seleccionando esta opción podrá redactar los cambios que considera oportunos, así como su justificación, mediante el cuadro de texto que se muestra en la Fig. 17.

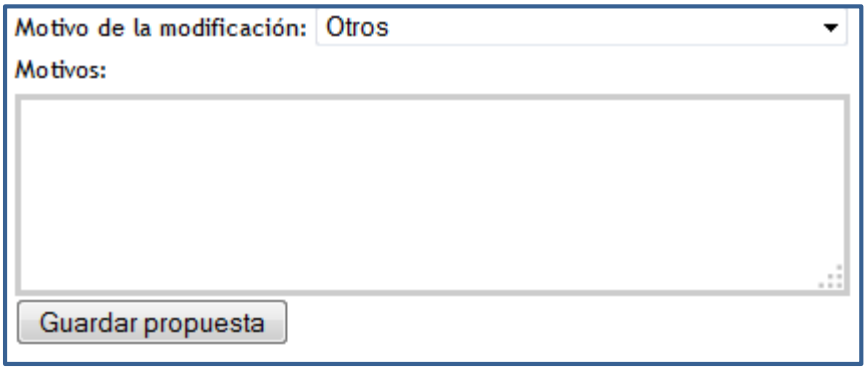


Fig. 17: Formulario "otros motivos"

Realizados los cambios que considere oportunos, el botón "Guardar propuesta" le permite almacenar provisionalmente la propuesta, actualizando la tabla de la parte superior con los nuevos datos, y continuar editando el resto de preguntas del mismo modo.

Una vez editadas todas las preguntas, al enviar las propuestas realizadas al sistema, éste le mostrará una página en la que podrá revisar las propuestas realizadas mediante una navegación basada en pestañas como la mostrada en la Fig. 18.

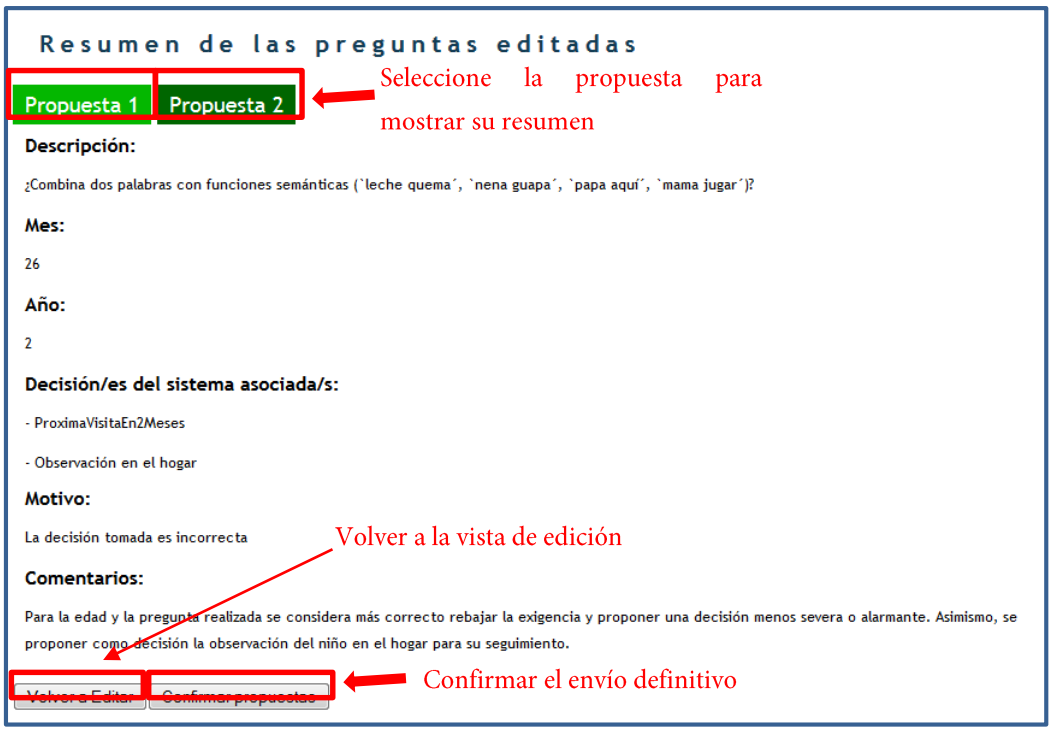


Fig. 18: Vista de resumen de las propuestas de cambio realizadas

Como se comentaba en el apartado anterior, una vez confirmado el envío de las propuestas al sistema no se podrán modificar de nuevo. Por ello, es importante la revisión de las mismas para, si se considera oportuno, volver a la página de edición y corregirlas.

Cuando considere que las propuestas son totalmente correctas, la confirmación del envío de las propuestas (Botón "Confirmar propuestas") supondrá su almacenamiento permanente.

4. Consulta de propuestas realizadas

Cualquier usuario puede consultar las propuestas que ha realizado utilizando la aplicación. Para ello, debe utilizar el acceso "Consultar histórico de propuestas" de la página de bienvenida.

Fig. 19: Página principal con funcionalidades habilitadas

Al pulsar sobre dicho acceso, se visualizará la siguiente página:

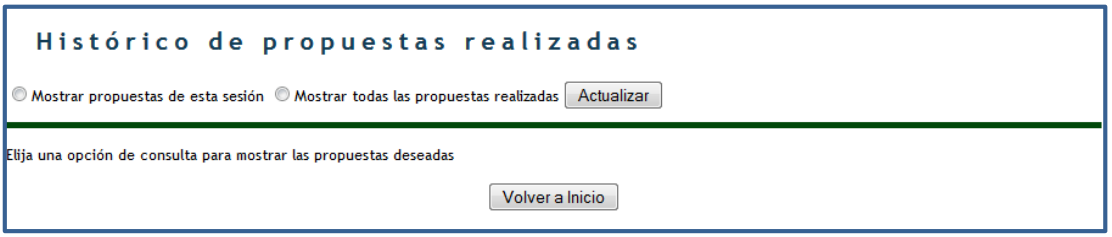
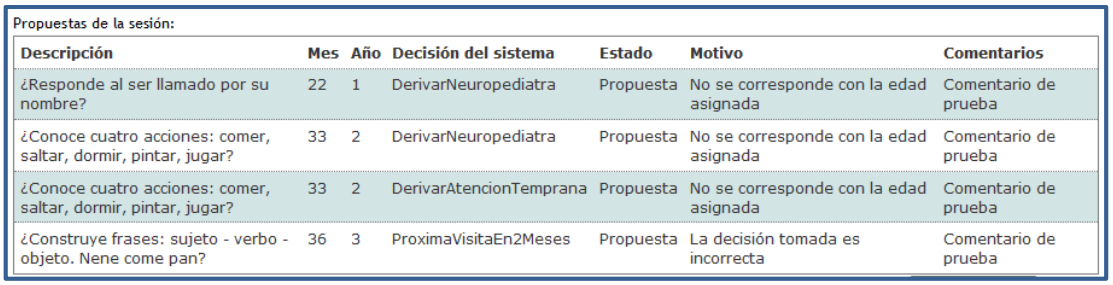


Fig. 20: Consulta de propuestas realizadas

Esta utilidad le permite filtrar su consulta en base a cuándo realizó dichas propuestas. Por un lado, le ofrece la posibilidad de mostrar las consultas realizadas en la sesión actual (desde que se autenticó en el sistema por última vez). Un ejemplo de esta consulta se muestra en la Fig. 21.



Propuestas de la sesión:

Descripción	Mes	Año	Decisión del sistema	Estado	Motivo	Comentarios
¿Responde al ser llamado por su nombre?	22	1	DerivarNeuropediatra	Propuesta	No se corresponde con la edad asignada	Comentario de prueba
¿Conoce cuatro acciones: comer, saltar, dormir, pintar, jugar?	33	2	DerivarNeuropediatra	Propuesta	No se corresponde con la edad asignada	Comentario de prueba
¿Conoce cuatro acciones: comer, saltar, dormir, pintar, jugar?	33	2	DerivarAtencionTemprana	Propuesta	No se corresponde con la edad asignada	Comentario de prueba
¿Construye frases: sujeto - verbo - objeto. Nene come pan?	36	3	ProximaVisitaEn2Meses	Propuesta	La decisión tomada es incorrecta	Comentario de prueba

Fig. 21: Consulta de propuestas realizadas en la sesión

Por otro lado, existe la posibilidad de mostrar todas las propuestas realizadas por el usuario. Si existe un gran volumen de propuestas se utilizarán varias páginas que le permitan navegar cómodamente, utilizando los botones de la parte inferior, tal y como muestra la Fig. 22.

Histórico de propuestas realizadas

Mostrar propuestas de esta sesión Mostrar todas las propuestas realizadas Actualizar

Histórico de propuestas:

Descripción	Mes	Año	Decisión del sistema	Estado	Motivo	Comentarios
¿Sabe hacer la correspondencia fonema grafema de varias palabras?	74	6	DerivarNeuropediatra	Propuesta	No se corresponde con la edad asignada	Comentario de prueba
¿Es capaz de interpretar dobles intenciones ? P.ej: juegos, trampas, bromas	72	6	DerivarAtencionTemprana	Propuesta	Pregunta mal formulada/ambigua	Comentario de prueba
¿Es capaz de interpretar dobles intenciones ? P.ej: juegos, trampas, bromas	72	6	DerivarNeuropediatra	Propuesta	Pregunta mal formulada/ambigua	Comentario de prueba
¿Reconoce metáforas: rubia como el oro?	72	6	ProximaVisitaEn3Meses	Propuesta	La decisión tomada es incorrecta	Comentario de prueba
¿Cuenta hasta diez?	61	5	DerivarNeuropediatra	Propuesta	Nueva propuesta	Comentario de prueba
¿Emplea `no´ en su lenguaje?	24	2	DerivarAtencionTemprana	Propuesta	Otros	Comentario de prueba
¿Emplea `no´ en su lenguaje?	24	2	DerivarNeuropediatra	Propuesta	Otros	Comentario de prueba
¿Responde al ser llamado por su nombre?	22	1	DerivarAtencionTemprana	Propuesta	No se corresponde con la edad asignada	Comentario de prueba

1ª página... página anterior... ...página siguiente ... última página Volver a Inicio

pag 2/3

Fig. 22: Consulta del histórico de propuestas

De no existir propuestas que cumplan el requisito seleccionado se mostrará un mensaje como el de la siguiente figura:

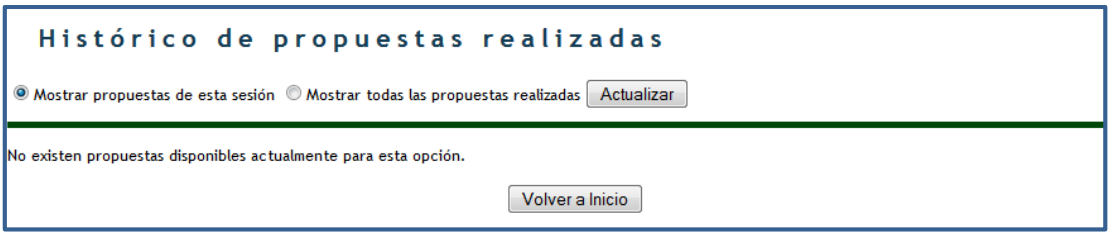


Fig. 23: Mensaje informativo si no existen propuestas para el tipo de consulta

5. Volver a la página de bienvenida desde cualquier página

Cuando se halle en cualquier página diferente a la página de bienvenida inicial podrá regresar a esta página inicial pulsando en el logo del sistema Galatea, como muestra la Fig. 24:

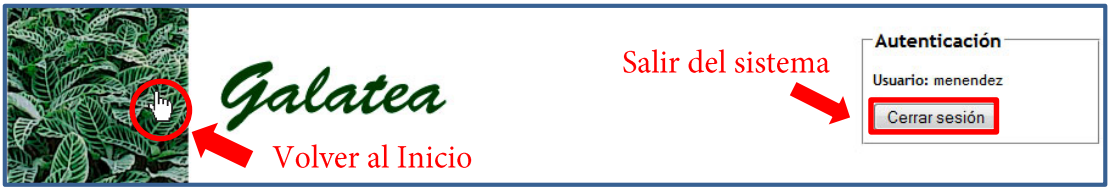


Fig. 24: Cabecera de Galatea

6. Salir del sistema

Puede salir del sistema en cualquier momento pulsando en el botón **Cerrar Sesión** disponible en todas las páginas de la aplicación (Fig. 24).

Se le solicitará que confirme que desea salir antes de cerrar su sesión y dirigirle a la página de entrada al sistema.